Optique

Project N^{o}:	**FP7-318338**
Project Acronym:	**Optique**
Project Title:	**Scalable End-user Access to Big Data**
Instrument:	**Integrated Project**
Scheme:	**Information & Communication Technologies**

Deliverable D4.2
Techniques for Ontology and Mapping Bootstrapping

Due date of deliverable:	(T0+24)
Actual submission date:	October 31, 2014

SEVENTH FRAMEWORK PROGRAMME

Start date of the project: **1st November 2012** Duration: **48 months**

Lead contractor for this deliverable: **UOXF**

Dissemination level: **PU** – Public

Final version

Executive Summary:
Techniques for Ontology and Mapping Bootstrapping

This document summarises deliverable D4.2 of project FP7-318338 (Optique), an Integrated Project supported by the 7th Framework Programme of the EC. Full information on this project, including the contents of this deliverable, is available online at `http://www.optique-project.eu/`.

More specifically, the present deliverable describes the designed and implemented *Techniques for Ontology and Mapping (O&M) Bootstrapping* corresponding to the Task T4.1.

Optique's O&M bootstrapping module allows to perform systems installation over relational databases. It offers several scenarios for installing the platform that combine *(i)* bootstrapping of ontologies and mappings from relational schemas, *(ii)* importing of existing ontologies in the platform via alignment or layering. O&M bootstrapper is tightly integrated with other Optique components and this allows to facilitate installation with mapping editing and ontology approximation. Moreover, O&M bootstrapper can encode in mappings information needed for provenance of query answers. We implemented the bootstrapper, integrated it in the Optique platform, and extensively evaluated on several database schemas including the ones provided by Statoil and Siemens. We currently work several challenging research directions that are tightly related to T4.1: bootstrapping of complex mappings and benchmarking. We presented our results on a number of international venues and to Statoil and Siemens users.

List of Authors

Ernesto Jiménez-Ruiz (UOXF)
Evgeny Kharlamov (UOXF)
Dmitriy Zheleznyakov (UOXF)
Ian Horrocks (UOXF)
Domenico Fabio Savo (UNIROMA1)
Valerio Santarelli (UNIROMA1)
Jose Mora (UNIROMA1)
Riccardo Rosati (UNIROMA1)
Marco Console (UNIROMA1)
Evgenij Thorstensen (UiO)
Dag Hovland (UiO)
Martin Giese (UiO)
Leif Harald Karlsen (UiO)
Daniel Lupp (UiO)
Martin Georg Skjæveland (UiO)
Johannes Trame (FOP)
Christoph Pinkel (FOP)
Thomas Hubauer (SIEMENS)
Mikhail Roshchin (SIEMENS)

Internal Reviewers

Martin Rezk (FUB)
Özgür L. Özçep (TUHH)

Contents

Chapter 1

Introduction

Building an ontology and connecting it to the data sources via mappings is a costly process, especially for large and complex databases. To aid this process, tools that can extract a preliminary ontology and mappings from the source schema play a critical role.

In order to ease the production of initial versions of the ontology and mappings, the Optique platform includes a *O&M bootstrapping* component that takes a set of database schemata as the input, and returns an ontology and a set of mappings that connect the terms occurring in the ontology to the schema elements.

The purpose of this document is to describe the designed and implemented *Techniques for Ontology and Mapping (O&M) Bootstrapping* corresponding to the Task T4.1 of WP4.

Challenges of the Work package. Within Optique, WP4 deals with the problems related to the management of the OBDA specification. The specification of an Ontology-Based Data Access (OBDA) system [14, 59] is a triple $\langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle$, where $\mathcal{O}$ is an ontology, providing a conceptual specification of the domain of interest, $\mathcal{S}$ is an intensional specification (schema) of a set of data sources, and $\mathcal{M}$ is a set of mapping assertions, i.e., expressions that specify the relationship between the ontology and the data sources, by means of queries over the ontology that are put in correspondence with queries over the data sources. OBDA systems crucially depend on the existence of suitable ontologies and mappings. Developing them from scratch in a Big Data scenario is likely to be expensive, thus, practical OBDA systems should support a (semi-)automatic creation of an initial ontology and set of mappings [35, 44].

Challenges of the Task. Task T4.1 has the goal of developing techniques and methodologies to facilitate the rapid deployment of the platform in new applications and application domains. Concretelly, this will lead to the development of the desired initial ontology and mappings.

Summary of Task Results. The Optique system, within its O&M Management system (see architecture in Figure 1.1), includes an *O&M bootstrapping* module that allows to perform systems installation over relational databases. O&M bootstrapper offers several scenarios for installing the platform that combine *(i)* bootstrapping of ontologies and mappings from relational schemas, *(ii)* importing of existing ontologies into the platform via alignment or layering. O&M bootstrapper is tightly integrated with other Optique components and this allows to facilitate installation with mapping editing and ontology approximation. Moreover, O&M bootstrapper can encode in mappings information needed for the provenance of query answers. We implemented the bootstrapper, integrated it in the Optique platform, and extensively evaluated on several database schemas including the ones provided by Statoil and Siemens. We currently work several challenging research directions that are tightly related to T4.1: bootstrapping of complex mappings and benchmarking. We presented our results on a number of international venues and to Statoil and Siemens users.

List of Achievement in Year 2.

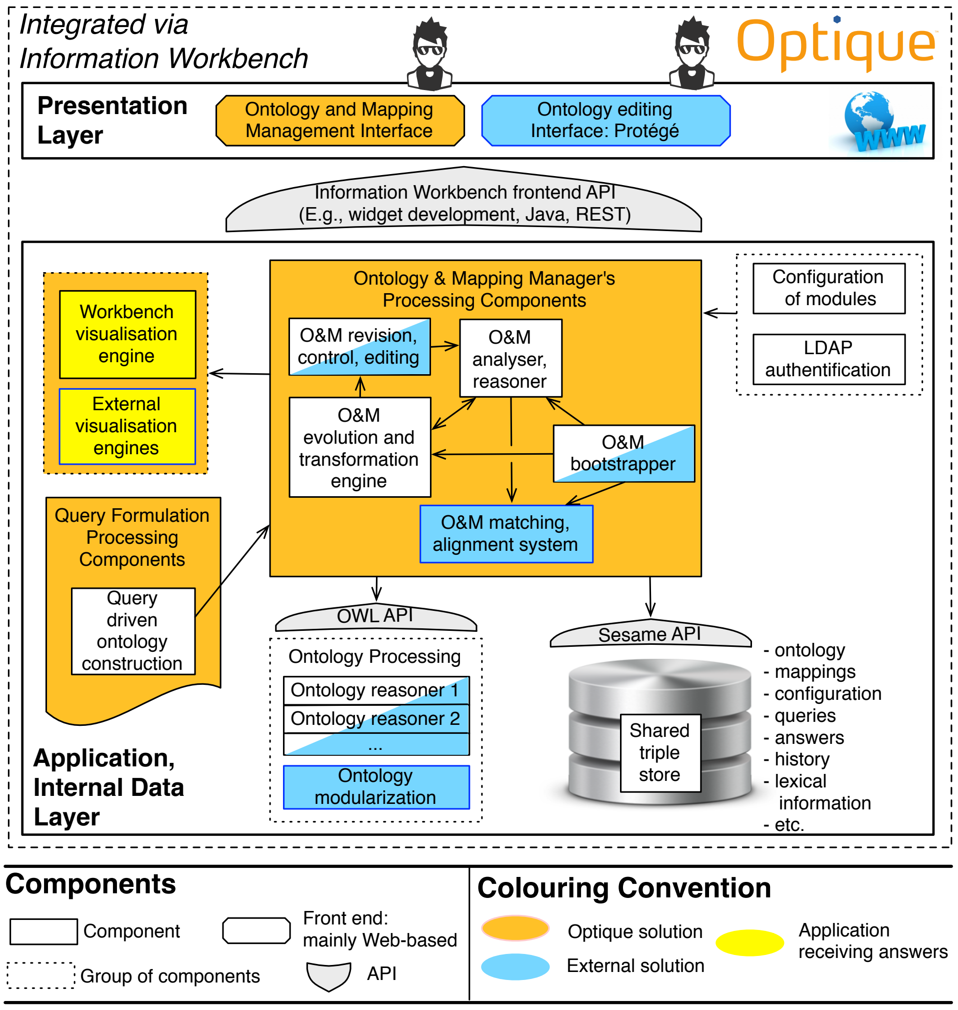

Integrated via Information Workbench
Optique
Presentation Layer
Ontology and Mapping Management Interface
Ontology editing Interface: Protégé
Information Workbench frontend API (E.g., widget development, Java, REST)
Workbench visualisation engine
External visualisation engines
Ontology & Mapping Manager's Processing Components
O&M revision, control, editing
O&M analyser, reasoner
O&M evolution and transformation engine
O&M bootstrapper
O&M matching, alignment system
Configuration of modules
LDAP authentification
Query Formulation Processing Components
Query driven ontology construction
OWL API
Ontology Processing
Ontology reasoner 1
Ontology reasoner 2
...
Ontology modularization
Sesame API
Shared triple store
- ontology
- mappings
- configuration
- queries
- answers
- history
- lexical information
- etc.
Application, Internal Data Layer
Components
Component
Front end: mainly Web-based
Group of components
API
Colouring Convention
Optique solution
External solution
Application receiving answers

Figure 1.1: Ontology and Mapping Management component of the Optique OBDA system

- O&M components of Year 1 implementation were improved and extended:
  - ontology alignment is improved with safety verification,
  - W3C test cases for the O&M bootstrapper were implemented,
  - the O&M bootstrapper is tightly integrated with the mapping editor,
  - installation GUI and wizards were improved,
- extensive experiments and evaluation of O&M bootstrapper were conducted on relational schemas from use-cases and other schemas,
- new techniques for bootstrapping of direct mappings, called *layering*, were developed, implemented, and integrated into the Optique platform,
- preliminary techniques for bootstrapping of complex mappings, i.e., involving select, project, and join relational operators were developed,
- preliminary techniques for embedding provenance in bootstrapped mappings were developed,
- work on benchmarking O&M bootstrapping approaches has been started.

Compliance with T4.1 Task Description and Relationship with Other Tasks and Work packages. Results of both Optique years in T4.1 match the goal of this task in the development of techniques to facilitate the rapid development of the platform in the new applications and application domains, in particular, in the development of an initial ontology and mappings. Techniques and software components developed within T4.1 are tightly connected with other tasks within WP4 and other work packages. In particular, the ontology approximation module developed within T4.2 as well as the mapping editor are integral parts of the bootstrapping workflows. Implementation and evaluation required in T4.4 was conducted in accordance with the task. The alignment component of the bootstrapper allows to integrate in the platform domain ontologies that are specifically developed for the purpose of query formulation and thus provides a bridge between T4.1 and WP3. The layering component allows to facilitate query driven ontology construction of WP3: one can extract semantic terms from users queries, layer them to the schema underlying the platform installation, and then integrate the terms in the platforms via ontology alignment.

Structure of the Deliverable. Chapter 2 presents some related work and motivates our decisions in the development of O&M bootstrapping techniques. Chapter 3 gives preliminaries and notation that we use in the deliverable. In Chapter 4 we presents main scenarios for platform installation. Chapter 5 presents our O&M bootstrapping techniques, which in particular support the above installation scenarios. More precisely, we start with a running example in Section 5.1, present mapping bootstrapping techniques in Section 5.2 present ontology bootstrapping techniques in Section 5.3, show how bootstrapped ontologies can be enhanced with external ones in Section 5.4, how to approximate ontologies in Section 5.5, how to integrate provenance in bootstrapped mappings in Section 5.6. Chapter 6 discusses hot to do analyse ontologies and mappings after they were bootstrapped. Chapter 7 discusses how the installation module is integrated in the platform. Chapter 8 explains how we conducted evaluation of the module. Finally, Chapter 9 reports on our ongoing work on benchmarking and bootstrapping of complex mappings.

Chapter 2

Related Work

In the literature one can find a broad range of approaches to bootstrap an ontology and mappings from a relational database schema. The interested reader may have a look at the following surveys [67, 72]. These approaches can be classified with respect to different aspects such as:*(i)* level of automation (i.e. manual, semi-automatic, automatic), *(ii)* type of mappings (i.e. complex or direct mappings), *(iii)* language of the bootstrapped mappings and the ontology, *(iv)* reuse of external vocabularies (e.g. domain ontologies or thesauri), and *(v)* purpose (e.g. OBDA, constraint validation, database integration, ontology learning).

RDFS and OWL 2 have been the most common languages for the bootstrapped ontology, although some systems have also used DLR-Lite based languages (e.g. [50]) and extensions based on SWRL (e.g. [31, 48]).

Many systems like D2RQ [10], SquirrelRDF [1], Ontop [62] and Mastro [18] used their own native language to define mappings before the R2RML specification [2] was completed.

The approaches in [49, 6, 50, 9, 63, 66, 60] present different solutions to create (automatic) *direct mappings* between the ontology and the relational database. Some of them (e.g., [50, 66, 60]), as our approach, closely follow the W3C "Direct Mapping of Relational Data to RDF" recommendation [3]; and only a few already focus on R2RML mappings (e.g., [66, 60]).

Systems like IncMap [58] present a more challenging approach where a domain ontology is directly mapped to the relational database schema in a semi-automatic process.

The approaches in [17, 16] uses ontology learning techniques to exploit the data and discover interesting patterns that can be included to enrich the ontology.

Finally, systems like Automapper [31], Relational.OWL [25] and ROSEX [23] complement the bootstrapped ontology with links to domain ontologies. The approach described in [63] also use background knowledge to infer new hierarchical relationships.

Although we could have reused one of the off-the-shelf bootstrapper we advocated for a custom implementation. As described above, most existing approaches focuse on languages of mappings and ontologies different from what we require for our OBDA solution (i.e. W3C standards). At the same time, our solution allows for a more fine-grained control over the different bootstrapping components and workflows in Optique, e.g., *(i)* communication with the Optique triple store to access the database schema and store the bootstrapped ontology and mappings; *(ii)* communication with the Optique APIs; *(iii)* use of ontology aligners to link the bootstrapped ontology with state of the art domain ontologies; *(iv)* use of ontology approximators to ensure that the bootstrapped is within the OWL 2 QL profile; *(v)* viualization and edition of the mappings after the bootstrapping.

In addition, our bootstrapper provides three different installation scenarios (see Section 4), which involve more or less manual intervention depending on the query requirements (i.e. the ontology may need to be extended and/or more sophisticated R2RML mappings may be required).

Chapter 3

Preliminaries

In this section we give some preliminaries about the resources, specifications and techniques we use in the deliverable.

3.1 Relational database

In this section we present some basic definitions related to relational data bases which will be used in the following sections. A relation database is composed by several schemas and each eschema by several tables.

We treat foreign key constraints as belonging to the attributes of a table, in the sense that they are preserved under projection and joins.

Definition 3.1.1 (Relation or Table) *A relation or table R is a set of attributes* $\mathsf{attr}(R) = \{a_1, \dots, a_n\}$ *together with a set of tuples over* $\mathsf{attr}(R)$*. Each attribute a has a type* $\mathsf{type}(a)$ *specifying the values it permits.*

Definition 3.1.2 (Projection) *Let R be a table, and $A = \{a_1, \dots, a_m\}$ a subset of* $\mathsf{attr}(R)$*. The projection of R onto A is the table $\pi_A(R) = \{t|_A \mid t \in R\}$, with set of attributes* $\mathsf{attr}(\pi_A(R)) = A$.

Definition 3.1.3 (Primary key) *Let R be a table. A primary key* $\mathsf{pk}(R)$ *for R is a declared subset of attributes $A \subseteq \mathsf{attr}(R)$ which uniquely identifies each tuple in R.*

Definition 3.1.4 (Superkeys) *Let R be a table. A set of attributes $A \subseteq \mathsf{attr}(R)$ is a* superkey *for R if for every pair of distinct tuples $t, t' \in R$, we have $t|_A \neq t'|_A$.*

Definition 3.1.5 (Joins on foreign keys) *We say that a table T_1 references another table T_2 whenever T_1 contains a foreign key that references T_2. We write* $\mathsf{fk}(T_1, T_2)$ *for this attribute of T_1, and* $\mathsf{refs}(T)$ *for the set of tables that reference T.*

If T_1 references T_2, we write $T_1 \bowtie_{\mathsf{fk}} T_2$ for the equijoin of T_1 and T_2 on the foreign key. Likewise, we write $T_1 \ltimes_{\mathsf{fk}} T_2$ for the left semijoin on the foreign key, that is, the set of tuples $\pi_{\mathsf{attr}(T_1)}(T_1 \bowtie_{\mathsf{fk}} T_2)$.

We also write $\mathsf{roj}_{\mathsf{fk}}(T_1, T_2)$ *for the right outer join of T_1 and T_2 on the foreign key.*

Definition 3.1.6 (Many to Many Tables) *In our setting, a relation R is considered as a many to many table if* $\mathsf{attr}(R) = \{a_1, a_2\} = \mathsf{pk}(R)$*, and a_1 and a_2 are foreign keys.*

3.2 Ontologies and the Web Ontology Language (OWL)

An ontology is usually referred to as a ‘conceptual model’ of (some aspect of) the world. It introduces the vocabulary of classes and properties that describe various aspects of the modelled domain. It also provides an explicit specification of the intended meaning of the vocabulary by describing the relationships between different vocabulary terms.

The Web Ontology Language (OWL), and its revision OWL 2 [20], are well known languages for ontology modeling in the Semantic Web. OWL ontologies are already being used in domains as diverse as bio-medicine, geology, agriculture, defence, and energy industry.

The formal underpinning of OWL 2 is provided by description logics (DLs) [7]—knowledge representation formalisms with well-understood formal properties. In this section, we very briefly summarise the basics of DLs, and refer the interested reader to [7, 20, 37] for further information.

DLs allow ontology developers to describe a domain of interest in terms of *individuals*, *atomic concepts* (usually called *classes* in OWL), and *roles* (also called *properties*). DLs also allow for *concept descriptions* (i.e., complex concepts) to be composed of atomic concepts and roles by providing a set of *concept constructors*. The DLs underlying OWL provide for intersection ($\sqcap$), union ($\sqcup$) and complement ($\neg$), as well as enumerated classes (called *oneOf* in OWL) and restricted forms of existential ($\exists$), universal ($\forall$) and cardinality restrictions ($\geq, \leq, =$) involving an atomic role R or its inverse R^-. A DL ontology $\mathcal{O}$ consists of a set of axioms. In the DLs underlying OWL it is possible to assert that a concept (or concept description) C is subsumed by (is a sub-concept of) D (written $C \sqsubseteq D$), or is exactly equivalent to D (written $C \equiv D$). It is also possible to assert subsumption of and equivalence between roles as well as to establish special constraints on roles (e.g., that a role should be interpreted as a transitive or as a functional relation).

DLs are equipped with a formal semantics, which enables the development of reasoning algorithms for answering complex queries about the domain. DLs, in fact, can be seen as decidable subsets of first-order logic, with individuals being equivalent to constants, concepts to unary predicates, and roles to binary predicates. As in the case of a first-order knowledge base, an interpretation $\mathcal{I}$ is a model of an ontology $\mathcal{O}$ (written $\mathcal{I} \models \mathcal{O}$) if $\mathcal{I}$ satisfies all the axioms in $\mathcal{O}$; $\mathcal{O}$ entails an axiom α (respectively an ontology $\mathcal{O}'$), written $\mathcal{O} \models \alpha$ ($\mathcal{O} \models \mathcal{O}'$), if $\mathcal{I} \models \alpha$ (respectively $\mathcal{I} \models \mathcal{O}'$) for every model $\mathcal{I}$ of $\mathcal{O}$. Finally $\mathcal{O}$ and $\mathcal{O}'$ are logically equivalent (written $\mathcal{O} \equiv \mathcal{O}'$) if $\mathcal{O} \models \mathcal{O}'$ and $\mathcal{O}' \models \mathcal{O}$.

3.2.1 The OWL 2 QL profile

The OWL 2 QL[1] profile is tailored for OBDA applications that aim at performing reasoning on top of very large volumes of data. One of the main motivations of this profile is to query data in a relational database using an ontology. Queries formulated over the ontology vocabulary are then translated into queries on the database using reasoning. The expressivity of OWL 2 QL is a bit restricted so that the ontology language underlying OWL 2 QL has the first-order (FO) rewritability property [15].

OWL 2 QL supports the following ontology axioms:

- subclass axioms
- class expression equivalence
- class expression disjointness
- inverse object properties
- property inclusion (not involving property chains)
- property equivalence
- property domain
- property range
- disjoint properties
- symmetric properties
- reflexive properties

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

- irreflexive properties
- asymmetric properties
- assertions other than individual equality assertions and negative property assertions

Class expression in subclass axioms and equivalence axioms are constrained as follows:

- **Subclass expression**: an atomic class, existential quantification where the class is limited to Top class, and existential quantification to a data range.
- **Superclass expression**: and atomic classes, intersection, existential quantification to a class, and existential quantification to a data range.

3.3 Ontology-to-Schema (OBDA) Mappings

Via ontology-to-schema mappings (or OBDA mappings or mappings for short) one can declaratively define how ontological terms are related to terms occurring in the relational schema. For example, mappings can be seen as view definitions of the following form that declare how to populate classes with objects—in OWL objects are represented with Uniform Resource Identifiers (URIs)—and to populate properties with object-object and object-value pairs:

$$\begin{aligned} \mathit{Class}(f_o(x)) &\leftsquigarrow \texttt{SQL}(x), \\ \mathit{Property}(f_o(x), f_o(y)) &\leftsquigarrow \texttt{SQL}(x,y), \\ \mathit{Property}(f_o(x), f_v(y)) &\leftsquigarrow \texttt{SQL}(x,y), \end{aligned}$$

where $\texttt{SQL}(x)$ and $\texttt{SQL}(x,y)$ are SQL queries with respectively one and two output variables,[2] and f_o, f_v are functions that 'cast' values returned by SQL into respectively objects, i.e, URIs, and values.[3] Classes are populated with URIs $f_o(x)$ computed from the values x returned by $\texttt{SQL}(x)$. Properties can relate two objects, e.g., by stating that Bob knows John, or assign a value to an object, e.g., by stating Bob's age is 25, and they are respectively populated with pairs of objects $f_o(x)$, $f_o(y)$ or pairs of an object $f_o(x)$ and value $f_v(y)$ computed from the values x and y returned by the SQL query.

Given a database and a set of mappings over it, one can execute SQL queries in the mapping definitions and populate the classes and properties of the mappings, thus creating a set of ontological facts. This process is usually referred to as virtual *materialisation* of the ontological facts defined by the mappings.

3.3.1 Direct Mapping Specification and R2RML language

A mapping is *direct* if it relates a table to a concept or an attribute to a property. In the O&M boostrapper we have closelly followed the W3C recommendation *A Direct Mapping of Relational Data to RDF*[4] which specifies a direct RDF Graph representation of the relational database.

For example, the table ***Person***(*id (PK), name, age, kwows (FK)*) with two rows (1, Bob, 25, 2) and (2, John, 30, 1) would be translated into the following triples:

- <http://base_uri/People/1> rdf:type <http://base_uri/People>
- <http://base_uri/People/1> <http://base_uri/People/name> Bob
- <http://base_uri/People/1> <http://base_uri/People/age> 25
- <http://base_uri/People/1> <http://base_uri/People/knows> <http://base_uri/People/2>

[2]Note that mappings may involve SQL queries with more than 2 variables.

[3] These functions should ensure coherent generation of URIs that respects primary and foreign keys.

[4]`http://www.w3.org/TR/rdb-direct-mapping/`

- <http://base_uri/People/2> rdf:type <http://base_uri/People>
- <http://base_uri/People/2> <http://base_uri/People/name> John
- <http://base_uri/People/2> <http://base_uri/People/age> 30
- <http://base_uri/People/2> <http://base_uri/People/knows> <http://base_uri/People/1>

Direct mappings are particular cases of the mappings that can be expressed in the R2RML standard.[5] R2RML is a W3C recommendation for expressing customized and expressive mappings from relational databases to RDF datasets. Intuitively, an R2RML mappings allows to map any valid SQL query or view (i.e. logical table) to a target vocabulary (e.g. ontology entities). R2RML mappings are RDF graphs and are typically stored in Turtle syntax.

R2RML mappings are composed by a set of *triples map*. A triples map specifies a rule for translating each row of a logical table to zero or more RDF triples. These rules are composed by a subject map (subjects often are IRIs that are generated from the primary key columns) and zero or more predicate-object maps.

For example, the *TriplesMap1* given below represents the direct R2RML mappings to translate the table given above into RDF data.

```
:TriplesMap1
  a rr:TriplesMap ;
  rr:logicalTable [ rr:tableName "People" ] ;
  rr:subjectMap [
    rr:template "http://base_uri/People/{id}" ;
    rr:class http://base_uri/People ;
  ] ;
  rr:predicateObjectMap [
    rr:predicate http://base_uri/name ;
    rr:objectMap [ rr:column "name" ]
  ] ;
  rr:predicateObjectMap [
    rr:predicate http://base_uri/age ;
    rr:objectMap [ rr:column "age" ]
  ] ;
  rr:predicateObjectMap [
    rr:predicate http://base_uri/knows ;
    rr:objectMap [
      rr:template "http://base_uri/People/{knows}"
    ]
  ] .
```

On the other hand *TriplesMap2* represents a more complex mapping where the logical table of the mappings is defined by a SQL query.

```
:TriplesMap2
  a rr:TriplesMap ;
  rr:logicalTable [ rr:sqlQuery "Select id FROM People WHERE age<26" ] ;
  rr:subjectMap [
    rr:template "http://base_uri/People/{id}" ;
    rr:class http://base_uri/YoungPeople ;
  ] .
```

3.4 Ontology-to-Ontology Alignments

In this section, we present the formal representation of ontology alignment and the notions of semantic difference, alignment coherence and conservativity principle violation.

[5] http://www.w3.org/TR/r2rml/

3.4.1 Representation of Ontology Alignments

Given two ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$ as input an ontology alignment system computes a set of correspondences or ontology alignments between the vocabularies of these ontologies.

Ontology alignments are conceptualised as 5-tuples of the form $\langle id, e_1, e_2, n, \rho \rangle$, with id a unique identifier, e_1, e_2 entities in the vocabulary or signature of the relevant input ontologies (*i.e.*, $e_1 \in \mathsf{Sig}(\mathcal{O}_1)$ and $e_2 \in \mathsf{Sig}(\mathcal{O}_2)$), n a confidence measure between 0 and 1, and ρ a relation between e_1 and e_2, typically subsumption (*i.e.*, e_1 is more specific than e_2), equivalence (*i.e.*, e_1 and e_2 are synonyms) or disjointness (*i.e.*, no individual can be an instance of both e_1 and e_2) [29].

RDF Alignment [24] is the main format used in the OAEI (Ontology Alignment Evaluation Initiative) campaign to represent mappings containing the aforementioned elements. Additionally, mappings are also represented as OWL 2 subclass, equivalence, and disjointness axioms [20]; mapping identifiers (id) and confidence values (n) are then represented as axiom annotations. Such a representation enables the reuse of the extensive range of OWL 2 reasoning infrastructure that is currently available. Note that alternative formal semantics for ontology mappings have been proposed in the literature (*e.g.*, [12]).

3.4.2 Semantic Consequences of the Integration

The ontology resulting from the integration of two ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$ via a set of alignments $\mathcal{A}$ may entail axioms that do not follow from $\mathcal{O}_1$, $\mathcal{O}_2$, or $\mathcal{A}$ alone.

In [39] three principles were proposed to minimize the number of potentially unintended consequences, namely: *(i) consistency principle*, the alignment should not lead to unsatisfiable classes in the integrated ontology, *(ii) locality principle*, the alignment should link entities that have similar *neighbourhoods*, *(iii) conservativity principle*, the alignment should not introduce new semantic relationships between concepts from one of the input ontologies.

In this deliverable, we focus on the consistency and conservativity principles.

Consistency principle

The consistency principle requires that the vocabulary in $\mathcal{O}_{\mathcal{U}} = \mathcal{O}_1 \cup \mathcal{O}_2 \cup \mathcal{A}$ be satisfiable, assuming the union of input ontologies $\mathcal{O}_1 \cup \mathcal{O}_2$ (without the alignments $\mathcal{A}$) does not contain unsatisfiable concepts. Thus $\mathcal{O}_{\mathcal{U}}$ should not lead to any axiom of the form $A \sqsubseteq \bot$, for any $A \in \Sigma = \mathsf{Sig}(\mathcal{O}_1 \cup \mathcal{O}_2)$.

Definition 3.4.1 (Alignment incoherence) *A set of alignments $\mathcal{A}$ is incoherent with respect to $\mathcal{O}_1$ and $\mathcal{O}_2$, if there exists a class A, in the signature of $\mathcal{O}_1 \cup \mathcal{O}_2$, such that $\mathcal{O}_1 \cup \mathcal{O}_2 \not\models A \sqsubseteq \bot$ and $\mathcal{O}_1 \cup \mathcal{O}_2 \cup \mathcal{A} \models A \sqsubseteq \bot$.*

An incoherent set of alignments $\mathcal{A}$ can be fixed by removing mappings from $\mathcal{A}$. This process is referred to as *alignments repair* (or repair for short).

Definition 3.4.2 (Alignment Repair) *Let $\mathcal{A}$ be an incoherent set of alignments w.r.t. $\mathcal{O}_1$ and $\mathcal{O}_2$. A set of alignments $\mathcal{R} \subseteq \mathcal{A}$ is a repair for $\mathcal{A}$ w.r.t. $\mathcal{O}_1$ and $\mathcal{O}_2$ iff $\mathcal{A} \setminus \mathcal{R}$ is coherent w.r.t. $\mathcal{O}_1$ and $\mathcal{O}_2$.*

A trivial repair is $\mathcal{R} = \mathcal{A}$, since an empty set of alignments is trivially coherent (according to Definition 3.4.1). Nevertheless, the objective is to remove as few alignments as possible. Minimal repairs are typically referred to in the literature as *mapping diagnoses* [52] — a term coined by Reiter [61] and introduced to the field of ontology debugging in [65].

Conservativity Principle

The conservativity principle (general notion) states that the integrated ontology $\mathcal{O}_{\mathcal{U}} = \mathcal{O}_1 \cup \mathcal{O}_2 \cup \mathcal{A}$ should not induce any change in the concept hierarchies of the input ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$. Note that we assume that the alignments $\mathcal{A}$ are coherent with respect to $\mathcal{O}_1$ and $\mathcal{O}_2$.

Definition 3.4.3 (Conservativity principle violations) *A set of alignments $\mathcal{A}$ violates the conservativity principle if $\mathcal{O}_1 \cup \mathcal{O}_2 \cup \mathcal{A} \models A \sqsubseteq B$ with A and B two entities in the signature of one of the input ontologies $\mathcal{O}_i$, and $\mathcal{O}_i \not\models A \sqsubseteq B$.*

A light variant of the conservativity principle was proposed in [69] (see Appendix F). This variant requires that the integrated ontology $\mathcal{O}_{\mathcal{U}}$ does not introduce new subsumption relationships between concepts from one of the input ontologies, unless they were already involved in a subsumption relationship or they shared a common descendant. The previous defintion should be extended with the conditions: *(i)* $\mathcal{O}_i \not\models B \sqsubseteq A$, and *(ii)* there is no C in the signature of $\mathcal{O}_i$ s.t. $\mathcal{O}_i \models C \sqsubseteq A$, and $\mathcal{O}_i \models C \sqsubseteq B$.

This variant of the conservativity principle follows the *assumption of disjointness* proposed in [64]. That is, if two atomic concepts A, B from one of the input ontologies are not involved in a subsumption relationship nor share a common subconcept (excluding $\bot$) they can be considered as disjoint. Hence, the conservativity principle can be reduced to the consistency principle, if the input ontologies are extended with sufficient disjointness axioms without causing logical conflicts.

3.5 Provenance

Provenance has been traditionally applied in the context of art or digital libraries as means to document the history of art objects or digital objects life cycles [32]. In our case we are concerned with provenance in OBDA systems for the main purpose of keeping the traceability of the data provided. OBDA systems provide access to data stored in data sources, usually databases, by generating new data that is part of an ontological model, usually assertions in an ABox, either materialized or virtual. By keeping the provenance relation between the newly generated data and the data used for its generation (in the data source) we can keep the traceability of the data. The traceability of the data provides access to the characteristics of the data that are dependent on the process used to obtain or derive this data up to its origin and context.

PROV-O PROV-O is an OWL2 ontology for the PROV Data Model [8], which allows to model provenance information in a semantically and automatically processable way. We will refer to URIs in the PROV-O namespace with the prefix "`prov:`". At its core, PROV-O models entities that are generated by some activity and attributed to some agent. Among the agents we can find software agents, as for example OBDA systems. Additionally, PROV-O allows modeling the "derivation" of some entities from other entities and the authorship of derived entities. For instance, when data is extracted from a database to derive individuals of an ontology we can specify that an individual was derived from (`prov:wasDerivedFrom`) the database or a part of it (e.g. a table) and that this individual was generated by (`prov:wasGeneratedBy`) the OBDA system or a part of it (e.g. a mapping assertion).

Provenance and OBDA mappings *Provenance in OBDA mappings* consists on maintaining the traceability provided by the provenance meta-information through the OBDA process. This means providing meta-information about the provenance of the data produced by the OBDA system (e.g. URIs, triples). This provenance meta-information will be related with the data sources to which the OBDA system is giving access and abstracting. For example: A particular triple may have been produced by a particular mapping assertion. This assertion would query a particular database and this could imply some other considerations like trust or quality (including monetary costs). All these considerations depend on the provenance of the information that the OBDA system produces. For clarity purposes, additionally to provenance in OBDA mappings we can consider two related problems: provenance of OBDA mappings and OBDA mappings for provenance data.

Provenance of OBDA mappings models the meta-information related with the provenance of a particular resource, in this case OBDA mappings. This type of provenance is a particular application of provenance models to a particular type of resource, in this case OBDA mappings. *OBDA mappings for provenance data* are mappings that provide access from an ontological view to a particular type of data, in this case provenance data. This type of OBDA mappings are a particular type of mappings that give access to a particular type of information that is stored in the database, in this case provenance information, but they are not qualitatively different from regular OBDA mappings.

3.6 Preliminaries for ontology approximation

Ontologies provide a conceptualization of a domain of interest which can be used for different objectives, such as providing a formal description of the domain of interest for documentation purposes, or providing a mechanism for reasoning upon the domain. For instance, they are the core element of the OBDA [14, 59] paradigm, in which the ontology is utilized as a conceptual view, allowing user access to the underlying data sources. With the aim to use an ontology as a formal description of the domain of interest, the use of expressive languages proves to be useful. If instead the goal is to use the ontology for reasoning tasks which require low computational complexity, the high expressivity of the language used to model the ontology may be a hindrance. In this scenario, the approximation of ontologies expressed in very expressive languages through ontologies expressed in languages which keep the computational complexity of the reasoning tasks low is pivotal. This motivates the study of the approximation of an ontology for OBDA applications, and thus, the study of approaches for approximating ontologies in very expressive languages with ontologies in languages, such as OWL 2 QL, that, characterized by low reasoning complexity, are suitable for query answering purposes.

Several approaches have recently dealt with the problem of approximating ontologies. These can roughly be partitioned in two types: *syntactic* and *semantic*. In the former, only the syntactic form of the axioms of the original ontology is considered, thus those axioms which do not comply with the syntax of the target ontology language are disregarded [74, 75]. This approach generally can be performed quickly and through simple algorithms. However, it does not, in general, guarantee soundness, i.e., to infer only correct entailments, or completeness, i.e., all entailments of the original ontology that are also expressible in the target language are preserved [56]. In the latter, the object of the approximation are the entailments of the original ontology, and the goal is to preserve as much as possible of these entailments by means of an ontology in the target language, guaranteeing soundness of the result. On the other hand, this approach often necessitates to perform complex reasoning tasks over the ontology, possibly resulting significantly slower. For these reasons, the semantic approach to ontology approximation poses a more interesting but more complex challenge.

3.6.1 Basic Definitions

Let Σ be a signature of symbols for individual (object and value) constants and predicates, i.e., concepts, value-domains, attributes, and roles. Let Φ be the set of all OWL 2 axioms over Σ.

With a slight abuse of notation, we say that an *ontology* over Σ is a finite subset of Φ. and that a *language* $\mathcal{L}$ over Σ is a set of ontologies over Σ. We call $\mathcal{L}$-*ontology* any ontology $\mathcal{O}$ such that $\mathcal{O} \in \mathcal{L}$. Moreover, we denote by $\Phi_{\mathcal{L}}$ the set of axioms $\bigcup_{\mathcal{O} \in \mathcal{L}} \mathcal{O}$.

We call a language $\mathcal{L}$ *closed* if $\mathcal{L} = 2^{\Phi_{\mathcal{L}}}$. It is easy to see that while OWL 2 QL is a closed language, OWL 2 is not. Indeed, OWL 2 imposes syntactic restrictions that concern the simultaneous presence of multiple axioms in the ontology (for instance, there exist restrictions on the usage of role names appearing in role inclusions in the presence of the role chaining constructor).

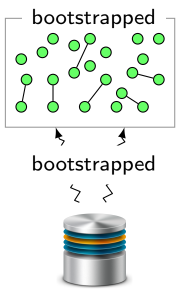
In what follows, we denote with $Mod(\mathcal{O})$ the set of models of $\mathcal{O}$. Moreover, given two ontologies $\mathcal{O}$ and $\mathcal{O}'$, we say that $\mathcal{O}$ and $\mathcal{O}'$ are logically equivalent if $Mod(\mathcal{O}) = Mod(\mathcal{O}')$.

Chapter 4

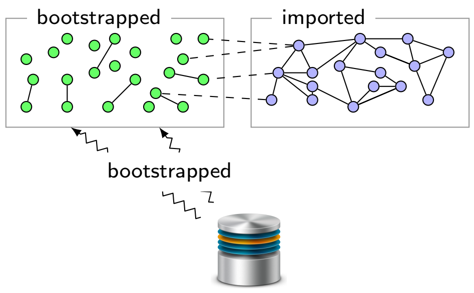
Installation Scenarios

As mentioned in Section 1 the ontology and mappings are the backbone of an OBDA system. Consequently, the problem of obtaining a suitable ontology and mappings has to be addressed in any implementation of the OBDA approach.

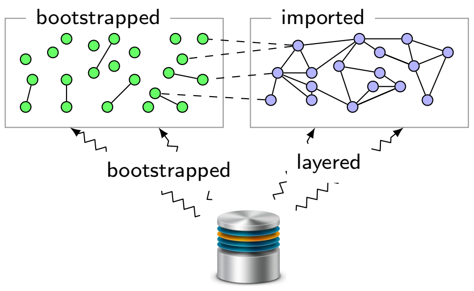
The Optique's O&M module provides semi-automatic support for the creation of the ontology and mappings by means of different installation scenarios that are schematically depicted in Figure 4.1.



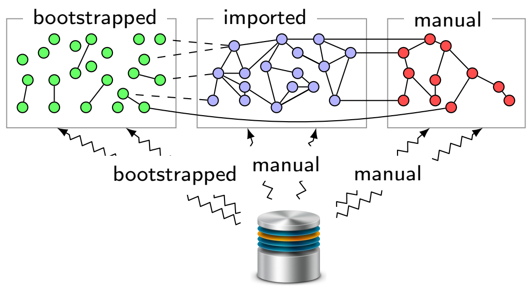
(a) Bootstrapping ontology and mappings



(b) Aligning bootstrapped and imported ontology



(c) Linking imported ontology directly to database



(d) Manual extensions of ontology and mappings

Figure 4.1: Installation scenarios

For example, the installation process usually starts with *bootstrapping*, i.e., automatic extraction of an ontology and mappings from the database (see Sections 5.2 and 5.3 for details) as depiected in Figure 4.1(a). Due to the nature of the automatic bootstrapping process, the bootstrapped ontology closely reflects the structure of the underlying database, and thus, it may not be ideal for query formulation support. To overcome this issue, the O&M bootstrapper allows importing of pre-existing external domain ontologies, whose vocabulary is preferred by the domain experts, and 'connect' them to the bootstrapped one via *ontology alignment* (see Section 5.4 for details) as depicted in Figure 4.1(b). Another possible installation scenario is to *layer* a pre-existing domain ontology directly over the database (see Section 6.1 for details), i.e., to 'connect' it to the database schema with semi-automatically generated mappings (Figure 4.1(c)). Both importing scenarios can address the case when there are several good domain ontologies available and they can serve as entry points to users with potentially different needs. Finally, one can manually edit and extend both ontology and mappings (see Section 6.3 for details) as depicted in Figure 4.1(d).

The Optique platform supports ontologies expressible in the OWL 2 QL profile of OWL 2 ontology language, which was specifically designed for efficient data access. Imported or layered ontologies that go beyond the expressivity of OWL 2 QL are approximated using the technique described in Section 5.5, which allows to project rich input OWL 2 ontologies into the OWL 2 QL profile.

Chapter 5

Bootstrapping Techniques

This chapter describes the designed techniques for the O&M boostrapper. The O&M boostrapper takes a set of database schemata as the input, and returns an OWL ontology and a set of (direct) R2RML mappings. Intuitively, mappings will relate tables and attributes in the schema to classes and properties in the ontology.

5.1 Running example

Figure 5.1 shows a fragment of a database schema based on the domain of the Optique project. The schema consist of 7 tables to store information about wellbores. For example, a *Wellbore* is given a name, has a content (e.g. GAS or OIL), belongs to a *Well*, has one *Operator* and is located in a *Field*. Additionally, the schema also stores information about types of wellbore. The examples in the subsequent sections will be based on this schema.

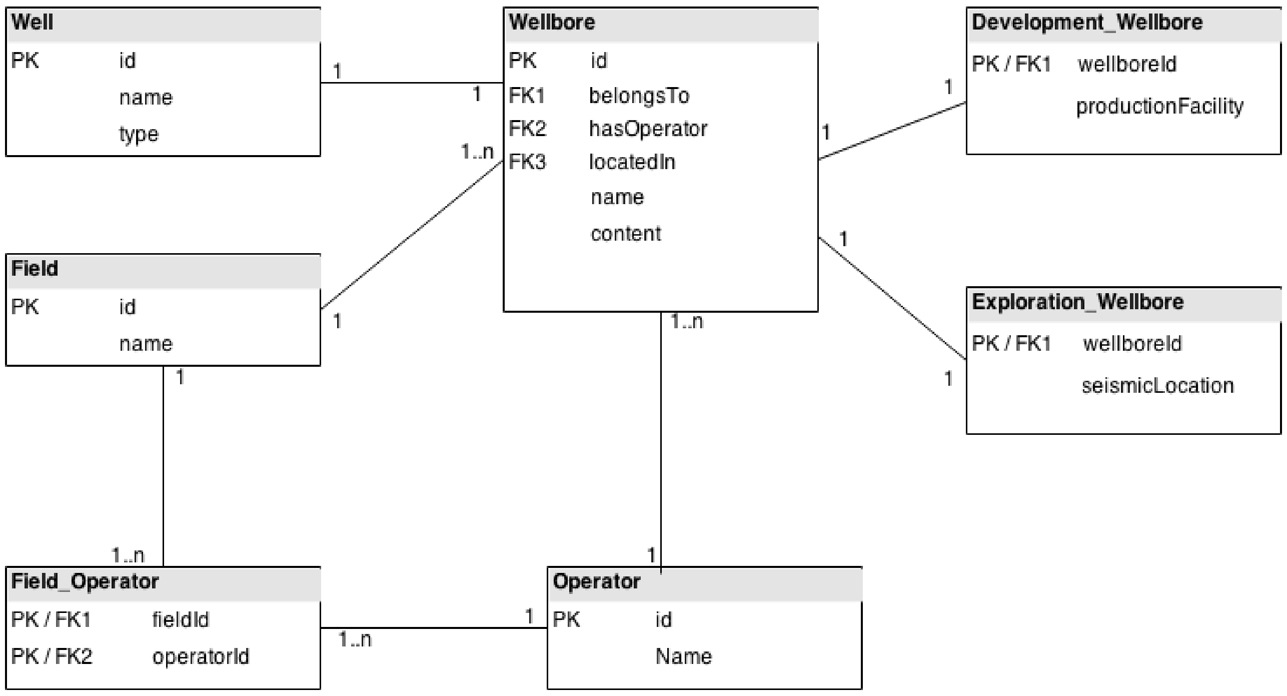


Figure 5.1: Fragment of a relational database schema

5.2 Bootstrapping of Mappings

The *bootstrapping* module follows the W3C direct mapping specification, as introduced in Section 3.3.1, in order to be compliant with the W3C standards. The W3C direct mapping specification provides a set of cases and guidelines[1] to generate R2RML mappings, for example, how to deal with missing primary keys, many-to-many tables, composite keys, etc. Appendix A summarizes the cases that we have considered in the mapping bootstrapper. Note that some types of mappings have not been considered since they require (directly or indirectly) knowledge about the data (i.e. use of complex logical tables) or they have been left for the extended boostrapper (i.e. null treatment). Some other types of mappings are considered as optional like the generation of blank nodes when the primary key is missing.

For example, consider the running example in Figure 5.1. The following three *triples maps* will be associated to the many-to-many table *Field_ Operator*. Note that, many-to-many tables are special cases within the direct mapping directives which does not imply a direct mapping from a table to an ontology class

```
:TriplesMap1
  a rr:TriplesMap ;
  rr:logicalTable [ rr:tableName "Field" ] ;
  rr:subjectMap [
    rr:template "http://base_uri/Field/{id}" ;
    rr:class <http://base_uri/Field> ;
  ] ;
  rr:predicateObjectMap [
    rr:predicate http://base_uri/name ;
    rr:objectMap [ rr:column "name" ]
  ] .

:TriplesMap2
  a rr:TriplesMap ;
  rr:logicalTable [ rr:tableName "Operator" ] ;
  rr:subjectMap [
    rr:template "http://base_uri/Operator/{id}" ;
    rr:class <http://base_uri/Operator> ;
  ] ;
  rr:predicateObjectMap [
    rr:predicate http://base_uri/name ;
    rr:objectMap [ rr:column "name" ]
  ] .

:TriplesMap3
  a rr:TriplesMap ;
  rr:logicalTable [ rr:tableName "Field_Operator" ] ;
  rr:subjectMap [
    rr:template "http://base_uri/Field/{fieldId}" ;
    rr:class <http://base_uri/Field> ;
  ] ;
  rr:predicateObjectMap [
    rr:predicate http://base_uri/hasOperator ;
    rr:objectMap [
      rr:template "http://base_uri/Operator/{operatorId}"
    ]
  ] .
```

Note that our ontology and mapping bootstrapping techniques minimises the effect of the so-called impedance mismatch problem [59], which is caused due to the fact that databases store data values (e.g strings, integers, etc.) while the ontology includes objects uniquely identified by URIs. We address the problem on the level of object generating functions discussed in Section 3.3. Our function respects primary

[1] http://www.w3.org/TR/rdb2rdf-test-cases/

and foreign keys and ensures that all the generated objects are unique and the same object is generated when required. The R2RML language provides mechanisms (i.e. templates) to implement the object generating functions. For example, the basic rule to generate the URIs of the subjects in a triples map is the following:

```
http://base_uri/Table_name/{pk1}/{pk2}/.../{pkn}
```

If the table does not contain any primery key the templare URI is created using all columns in the table:

```
http://base_uri/Table_name/{col1}/{col2}/.../{coln}
```

Note that, many-to-many tables are special cases where no new subject URIs are defined.

When dealing with foreign keys, the URI of the object in the triples map associated to the referencing table should be generated according to the URIs of the subjects of the referenced table. e.g.:

```
http://base_uri/Referenced_Table/{fk_col1}/{fk_col2}/.../{fk_coln}
```

In our example given above, the subjects and objects of the *TriplesMap3* uses the same URIs as the subjects in *TriplesMap1* and *TriplesMap2*, respectively.

5.2.1 Layering an Existing Ontology

The W3C direct mapping specification does not introduce specific restrictions on the used ontology. The R2RML mappings, for example, only require to reference the vocabulary (i.e. entity URI) of an exixting ontology. Thus, the referenced ontology entities can come from a pre-existing external ontology and/or from the results of the (ontology) bootstrapping (see Section 5.3).

For example, consider the running example in Figure 5.1. The table *Field* could be mapped to the boostrapped ontology entity `http://base_uri/Field` and to the entity `http://sws.ifi.uio.no/vocab/npd-v2#Field` from the Norwegian Petroleum Directorate (NPD) ontology [68] as in the following mapping.

```
:TriplesMap4
  a rr:TriplesMap ;
  rr:logicalTable [ rr:tableName "Field" ] ;
  rr:subjectMap [
    rr:template "http://base_uri/Field/{id}" ;
    rr:class <http://base_uri/Field>,<http://sws.ifi.uio.no/vocab/npd-v2#Field>;
  ] .
```

The (automatic) layering of a pre-existing ontology (i.e., to 'connect' it to the database schema with direct mappings), requries the discovery of lexical correspondences between table names and ontology class names, and between attribute names and ontology property names. Section 6.1 presents a semi-automatic process to validate these discovered mappings.

This (installation) scenario can address the case when there are several good ontologies available and they can serve as entry points to data for users with potentially different needs.

5.3 Bootstrapping of Ontologies

The W3C direct mapping specification does not introduce specific restriction on the axioms of the bootstrapped ontology. As discussed in Chapter 2 there are several solutions in the literature to infer axioms from the database schema constraints. Intuitively, a basic boostrapper would do the following: *(i)* each relation or table R is translated into an OWL class; *(ii)* each attribute a not involved in a foreign key is translated into an OWL datatype property; and *(iii)* each foreign key is translated in an OWL object property. Next we present our solution.

- **Vocabulary extraction**

- **Naming conventions.** As in the R2RML mappings, the use of a proper naming convention is very important. Table and attribute names from a relational database schema are translated into URIs which are the unique identifiers for OWL entities (i.e. classes, data properties, object properties and instances). $\mathsf{uri}(R)$ and $\mathsf{uri}(a)$ denote the URI of a relation R and an attribute a, respectively. The URIs $\mathsf{uri}(R)$ are created using a base URI (i.e. `http://base_uri/`) and the name of the table (i.e. `http://base_uri/Field`), `uri:Field` or `field` for short). The URIs $\mathsf{uri}(a)$ are created using the base URI, the name of the table of the attribute a, and the name of the attribute a.
- **OWL classes.** Each relation or table R (apart from many-to-many tables) is translated into an OWL class. For example, in our running example depicted in Figure 5.1, the boostrapper would create 6 classes.
- **OWL data properties.** Each attribute not involved in a foreign key is translated into an OWL data property. For example, the attribute *name* in the *Field* table in Figure 5.1 is represented with a data property with URI `field:name`.
- **OWL object properties.** Each foreign keys is translated into a OWL object property. For example, the foreign key attribute *locatedIn* in the relation *Wellbore* is represented with the object property with URI `wellbore:locatedIn`

• **Propagation of constraints**

- **Domain axioms.** Each data and object property is associated with an OWL property domain axiom. For example, the property `field:name` representing the attribute *name* in the table *Field* (see Figure 5.1) is associated the class `uri:Field` as domain. Thus the axiom *DataPropertyDomain*(field:name uri:Field) is added to the boostrapped ontology.
- **Data Range axioms**. The type of an attribute $\mathsf{type}(a)$ will be represented as OWL data range axiom. For example, $\mathsf{type}(name)$ in table *Field* (see Figure 5.1) is represented as the ontology axiom *DataRange*(field:name xsd:string).
- **Object Range axioms**. Foreign keys link one table with another, thus the range of an object property is the destination table of the source foreign key. For example, the foreign key *locatedIn* in the relation *Wellbore* (see Figure 5.1) points to the relation *Field*, and hence the OWL object property range axiom *ObjectPropertyRange*(wellbore:locatedIn uri:Field) is added to the boostrapped ontology.
- **Existential restriction.** Non nulleable attributes a are translated into OWL existential restrictions. For example, in our running example, the *Wellbore* attributes *name* and *hasOperator* are declared as *not nulleable.* Hence the boostrapped ontology is extended with the following OWL axioms: $\mathsf{uri{:}Wellbore} \sqsubseteq \exists \mathsf{wellbore{:}name.xsd{:}String}$ and $\mathsf{uri{:}Wellbore} \sqsubseteq \exists \mathsf{wellbore{:}hasOperator.uri{:}Operator}$.

• **Dealing with many-to-many tables.** Many-to-many tables requires a special attention both in the mapping and ontology generation. These tables are required by database management systems that only support one-to-many relationships. However, in a higher level representation such as an ontology, these *junction relationships* can be avoived by adding direct relationships between the referenced tables in the many-to-many table. Concretelly, our boostrapper enriches the ontology with the following information:

- **Object properties.** Two new OWL object properties are created. One represents the relationship between the referenced table T_1 to the referenced table T_2, while the second property represents the inversere relationship.
- **Domain and range axioms.** The new object properties are associated a domain and a range, which are the corresponding OWL classes of the referenced tables T_1 and T_2 in the many-to-many relation, respectively.

– **Inverse property axioms.** The two new properties are explicitly declared as inverse.

For example, in the running example depicted in Figure 5.1, the many-to-many table *Field_ Operator* leads to the creation of the object properties *hasOperator* and *isOperatorOf*, one inverse of the other. The domain and range of *hasOperator* are *Field* and *Operator*, repsectively. While *isOperatorOf* has *Operator* as domain and *Field* as range.

5.3.1 Adding hierarchy to the ontology classes

The O&M bootstrapper performs a basic discovery of potential subclass relationships between the classes in the ontology. This *automatic* discovery may require a manual assessment to validate the new subclass relationships. For every pair of tables R and T:

(i) if $\mathsf{pk}(R) = \mathsf{fk}(R, T)$ then the OWL class axiom *SubClassOf*$(\mathsf{uri(R)}\ \ \mathsf{uri(T)})$ is suggested. For instance, in the example schema given in Figure 5.1, the *DevelopmentWellbore* table's primary key is also a foreign key pointing to the *Wellbore* table. It turns out to be an effective heuristic in such cases to add a class inclusion axiom between these classes (i.e. $\mathsf{DevelopmentWellbore} \sqsubseteq \mathsf{Wellbore}$).

(ii) if $\mathsf{attr}(R) \subset \mathsf{attr}(T)$ then the OWL class axiom *SubClassOf*$(\mathsf{uri(T)}\ \ \mathsf{uri(R)})$ is suggested.

(iii) if $\mathsf{attr}(R) \cap \mathsf{attr}(T)$ is a superkey for both R and T then we suggest a new class $\mathsf{uri}(S_{RT})$ and the axioms *SubClassOf*$(\mathsf{uri(R)}\ \ \mathsf{uri}(\mathsf{S_{RT}}))$ and *SubClassOf*$(\mathsf{uri(T)}\ \ \mathsf{uri}(\mathsf{S_{RT}}))$ to be added to the ontology.

5.3.2 Dealing with multiple properties with the same name

The O&M bootstrapper associates to each attribute a unique URI. For example, cosider the schema given in Figure 5.1, the attribute *name* in the relation *Field* and the attribute with the same name, i.e., *name*, in the relation *Operator* get different URIs in the bootstrapped ontology. In our experiments we observed that in some cases the introduction of different URIs for the same attribute names was hightly redundant. There are several candidate solution to avoid such redundancy:

Compact attribute names If the O&M bootstrapper works under the compact attribute names regime, then for compatible (i.e. same type and name) attributes will give the same URI. Hence, in the ontology, the domain of these attributes will be composed by a union of classes. In the case of foreign keys, the range of the derived object properties in the ontology will also be an union of classes. Note that, disjunction is outside the OWL 2 QL profile. Alternatively, instead of adding multiple domain and ranges, we can create existential restriction to be attached to each domain class. For example, for the attribute *name* in Relation *Wellbore* we could add the axiom $\mathsf{uri{:}Wellbore} \sqsubseteq \exists\mathsf{uri{:}name.xsd{:}String}$. This solution can be problematic if the attribute is nulleable. In Section 5.3.3 a complementary solution using annotation properties is also proposed.

Adding a super property A simpler solution involves the addition of a super-property that groups compatible properties. For example, the properties *well:name*, *wellbore:name*, *field:name* and *operator: name* are grouped under the new property *uri:name*.

5.3.3 Annotation Schema for the Query Formulation Interface

The Optique Visual Query Formulation Interface (OptiqueVQS) is driven by the information available in the (bootstrapped) ontology. We observed in our user study about the OptiqueVQS [71], that a purely axiom driven query interface suffers from important practical limitations, e.g., it does not allow users to set specific data values in queries, e.g., company/operator names. To address this issue we have enriched the ontology with annotations.[2] For example, we precomputed values that are frequently used, rarely changed,

[2]Note that, lists of values and numerical ranges in an OWL data property range fall outside OWL 2 QL, and thus it should be encoded as non logical axioms

and from relatively small domains; this includes names of companies and oilfield, geolocations, temporal information, ranges of numerical values, e.g., min/max possible depth of wellbores. The OptiqueVQS has also been extended with a module to automatically customize the query interface by displaying data values as pre-populated dropdown lists and range sliders.

In addition, as mentioned in Section 5.3.2, multiple property domains and ranges (i.e. dijunction of classes as domain and ranges) is not permitted in OWL 2 QL. Thus an OWL 2 QL approximator as the one to be presented in Section 5.5 will approximate or remove these axioms. However, although this information may not have a crucial impact in the rewriting process, it does have an important role in the OptiqueVQS, as for the list of values and numerical ranges in an OWL data property range. Hence, in order to be able to keep this non OWL 2 QL information, we have added annotations to the ontology about the multiple domains and ranges.

The annotations have been defined using the following annotation schema based on OWL 2 annotations axioms:[3]

- `http://eu.optique.ontology/annotations#geoLocation`: this annotation property is used to annotate class with geo-location information such as *fields* or *wellbores*.
- `http://eu.optique.ontology/annotations#temporal`: this annotation property is used to annotate classes with temporal information such as *events* or *measurements*.
- `http://eu.optique.ontology/annotations#data_values`: this annotation property annotates data properties with specific data range values such as *company names* or *field names*.
- `http://eu.optique.ontology/annotations#range_class`: this annotation property annotates object properties with class ranges. For example, the property *uri:hasOperator* is annotated with *Operator* as range class.
- `http://eu.optique.ontology/annotations#domain_class`: this property annotates properties with domains. For example, the property *uri:name* is annotated with *Operator* and $Field$ as domain classes. Additionally, this annotation property will also annotate annotation axioms using $range\_class$ or $data\_values$. That is, the annotation axiom for $data\_values$ representing the list of company names is annotated with *Company* as domain class.
- `http://eu.optique.ontology/annotations#min_values`: this annotation property indicates the minimum value in a range of numerical values for a data property.
- `http://eu.optique.ontology/annotations#max_values`: this annotation property indicates the maximum value in a range of numerical values for a data property.
- `http://eu.optique.ontology/annotations#hidden`: this annotation will indicate if a data property should be considered for visualization in the query formulation interface.

5.4 Enhancing Bootstrapping with External Ontology

The boostrapped ontology, although it has been enriched with logical axioms as described in Section 5.3, it will still usually be too close to the source schema. It is however increasingly often the case that a high quality ontology of (parts of) the domain already exists, that captures the domain experts' vocabulary better than the directly mapped ontology.

When such a high quality ontology is available, the bootstrapping component allows importing it and using it in the bootstrapping process (i.e. alignment of the directly mapped and the imported ontologies). Special care needs to be taken to avoid introducing unwanted consequences: for instance the bootstrapper will avoid adding alignment axioms that would lead to inconsistencies, or faulty consequences like $\mathsf{Well} \sqsubseteq \mathsf{WellBore}$

[3] `http://www.w3.org/TR/owl2-new-features/#Extended_Annotations`

Algorithm 1: Algorithm to detect and solve conservativity principle violations

```
1  Input: $\mathcal{O}_1$, $\mathcal{O}_2$: input ontologies; $\mathcal{A}$: (coherent) input mappings;
2  Output: $\mathcal{A}'$: output mappings; $\mathcal{R}^{\approx}$: approximate repair; $disj$: number of disjointness rules
3  $\langle \mathcal{O}'_1, \mathcal{O}'_2 \rangle := \mathsf{ModuleExtractor}(\mathcal{O}_1, \mathcal{O}_2, \mathcal{A})$
4  $\langle \mathcal{P}_1, \mathcal{P}_2 \rangle := \mathsf{PropositionalEncoding}(\mathcal{O}'_1, \mathcal{O}'_2)$
5  $SI_1 := \mathsf{StructuralIndex}(\mathcal{O}'_1)$, $SI_2 := \mathsf{StructuralIndex}(\mathcal{O}'_2)$
6  $SI_{\mathcal{U}} := \mathsf{StructuralIndex}(\mathcal{O}'_1 \cup \mathcal{O}'_2 \cup \mathcal{A})$
7  $\langle \mathcal{P}^d_1, disj_1 \rangle := \mathsf{DisjointAxiomsExtension}(\mathcal{P}_1, SI_1, SI_{\mathcal{U}})$
8  $\langle \mathcal{P}^d_2, disj_2 \rangle := \mathsf{DisjointAxiomsExtension}(\mathcal{P}_2, SI_2, SI_{\mathcal{U}})$
9  $\langle \mathcal{A}', \mathcal{R}^{\approx} \rangle := \mathsf{MappingRepair}(\mathcal{P}^d_1, \mathcal{P}^d_2, \mathcal{A})$     ◇ See Algorithm 2 in [30] (Appendix G)
10 return $\langle \mathcal{A}', \mathcal{R}^{\approx}, disj_1 + disj_2 \rangle$
```

or $\mathsf{Well} \sqsubseteq \bot$ that are not supported by the domain ontology. Note that these unintended consequences may hinder the usefulness of the generated ontology alignments since they may affect the quality of the results when performing OBDA queries over the vocabulary of the aligned ontology.[4]

5.4.1 Ontology Alignment

Domain ontologies may be complex ontologies describing hundred or even thousands of classes; as a result, computing alignments between the directly mapped and the imported ontologies may be infeasible without suitable tool support. Thus, the alignment process in the O&M module relies on the LogMap system [38, 40]. LogMap is a highly scalable ontology matching system that discovers ontology-to-ontology alignments, e.g. class inclusionss, between the vocabularies of the input ontologies. LogMap can efficiently match semantically rich ontologies containing hundreds of thousands of classes, and it is one of the few tools that integrates reasoning techniques to minimise the number of unintended logical errors.

For use cases requiring very accurate alignments, LogMap also supports user interaction during the alignment process in order to keep the number of wrong correspondences to a minimum. Thi is an important step in the validation of the bootstrapped ontology (see Section 6.2).

LogMap obtained very good results in the last Ontology Alignment Evaluation Initiative[5] (OAEI) and it was among the top 3 (out of 23) systems participating. The OAEI is an annual international campaign for the systematic evaluation of ontology alignment systems [5]. Appendixes D and E provide an overview of the techniques implemented in LogMap and the results obtained in the OAEI 2013 and 2014 campaigns.

5.4.2 Alignment Repair

LogMap also implements automatic repair techniques [69, 30] (see Appendixes F and G) to avoid unintended logical consequences (as the ones described above) after the alignment. As introduced in Section 3.4.2 these logical errors are characterised as violations of the consistency and conservativity principles.

In this section we focus on the implemented methods to detect and repair violations of the conservativity principle. As introduced in Section 3.4.2, following our notion of conservativity, we have reduced the problem of detecting and solving conservativity principle violations to an alignment (incoherence) repair problem.

Algorithm 1 shows the pseudocode of the implemented method. The algorithm accepts as input two OWL 2 ontologies, $\mathcal{O}_1$ and $\mathcal{O}_2$, and an alignment $\mathcal{A}$ which are coherent[6] w.r.t. $\mathcal{O}_1$ and $\mathcal{O}_2$. The problem size is reduced by extracting two locality-based modules [22] ($\mathcal{O}'_1$ and $\mathcal{O}'_2$) using the entities involved in the alignment $\mathcal{A}$ as seed signatures (line 3, Algorithm 1). The output is the number of added disjointness during the process $disj$, the repaired alignment $\mathcal{A}'$, and an approximate repair $\mathcal{R}^{\approx}$ s.t. $\mathcal{A}' = \mathcal{A} \setminus \mathcal{R}^{\approx}$. The repair $\mathcal{R}^{\approx}$ aims at solving most of the basic violations of $\mathcal{A}$ w.r.t. $\mathcal{O}_1$ and $\mathcal{O}_2$. We next describe the techniques used in each step.

[4]Section 3 in the paper in Appendix [69] presents an example about the potential negative impact of unidesired consequences.
[5]`http://oaei.ontologymatching.org/`
[6]Note that $\mathcal{A}$ may be the result of a prior alignment (incoherence) repair process.

Propositional Horn Encoding. The modules $\mathcal{O}'_1$ and $\mathcal{O}'_2$ are encoded as the Horn propositional theories, $\mathcal{P}_1$ and $\mathcal{P}_2$ (line 4 in Algorithm 1). This encoding includes rules of the form $A_1 \wedge \ldots \wedge A_n \rightarrow B$. For example, the concept hierarchy provided by an OWL 2 reasoner (*e.g.*, [55, 42]) will be encoded as $A \rightarrow B$ rules, while the explicit disjointness relationships between concepts will be represented as $A_i \wedge A_j \rightarrow \mathsf{false}$. Note that the input alignment $\mathcal{A}$ can already be seen as a set of propositional implications.

Structural Index. The concept hierarchies provided by an OWL 2 reasoner (excluding $\bot$) and the explicit disjointness axioms of the modules $\mathcal{O}'_1$ and $\mathcal{O}'_2$ are efficiently indexed using an interval labelling schema [4] (line 5 in Algorithm 1). This structural index allows us to answer many entailment queries over the concept hierarchy as an index lookup operation (*i.e.*, without the need of an OWL 2 reasoner). For example, to check if two propositional variables (*i.e.*, ontological classes) are disjoint $(\mathsf{areDisj}(SI, A, B))$, if they keep a sub/super-class relationship $(\mathsf{inSubSupRel}(SI, A, B))$, or if they share a descendant $(\mathsf{shareDesc}(SI, A, B))$.

Disjointness Axioms Extension. In order to reduce the conservativity problem to an alignment incoherence repair problem, following the notion of *assumption of disjointness*, we need to automatically add sufficient disjointness axioms into each module $\mathcal{O}'_i$. However, additional disjointness axioms δ may lead to unsatisfiable classes in $\mathcal{O}'_i \cup \delta$.

For avoiding an extensive use of a costly OWL 2 reasoner, our method exploits the propositional encoding and structural indexing of the input ontologies. Thus, checking for unsatisfiabilities introduced by candidate disjointness axioms in $\mathcal{O}'_i \cup \delta$ is restricted to the Horn propositional case. We have implemented an algorithm to extend the propositional theories $\mathcal{P}_1$ and $\mathcal{P}_2$ with disjointness rules of the form $A \wedge B \rightarrow \bot$ (see lines 7-8 in Algorithm 1). This algorithm guarantees that, for every propositional variable A in the extended propositional theory $\mathcal{P}^d_i$ (with $i \in \{1, 2\}$), the theory $\mathcal{P}^d_i \cup \{true \rightarrow A\}$ is satisfiable. Note that this does not necessarily hold when considering the OWL 2 ontology modules, $\mathcal{O}'_1$ and $\mathcal{O}'_2$, as discussed above.

Repair. Line 9 of Algorithm 1 uses the mapping (incoherence) repair algorithm presented in [38] for the extended Horn propositional theories $\mathcal{P}^d_1$ and $\mathcal{P}^d_2$, and the input mappings $\mathcal{A}$. The mapping repair process exploits the Dowling-Gallier algorithm for propositional Horn satisfiability [28] and checks, for every propositional variable $A \in \mathcal{P}^d_1 \cup \mathcal{P}^d_2$, the satisfiability of the propositional theory $\mathcal{P}_A = \mathcal{P}^d_1 \cup \mathcal{P}^d_2 \cup \mathcal{A} \cup \{true \rightarrow A\}$. Satisfiability of $\mathcal{P}_A$ is checked in worst-case linear time in the size of $\mathcal{P}_A$, and the number of Dowling-Gallier calls is also linear in the number of propositional variables in $\mathcal{P}^d_1 \cup \mathcal{P}^d_2$. The algorithm records the *conflicting* mappings involved in the unsatisfiability, which will be considered for the subsequent repair process. The unsatisfiability will be fixed by removing some of the identified mappings. In case of multiple options, mapping confidence will be used as a differentiating factor.

Algorithm 1 gives as output the number of added disjointness rules $disj$, a set of mappings $\mathcal{A}'$, and an (approximate) repair $\mathcal{R}^{\approx}$ such that $\mathcal{A}' = \mathcal{A} \setminus \mathcal{R}^{\approx}$. $\mathcal{A}'$ is coherent w.r.t. $\mathcal{P}^d_1$ and $\mathcal{P}^d_2$. Furthermore, the propositional theory $\mathcal{P}_1 \cup \mathcal{P}_2 \cup \mathcal{A}'$ does not contain any conservativity principle violation w.r.t. $\mathcal{P}_1$ and $\mathcal{P}_2$. However, our incomplete encoding cannot guarantee that $\mathcal{O}'_1 \cup \mathcal{O}'_2 \cup \mathcal{A}'$ does not contain violations w.r.t. $\mathcal{O}'_1$ and $\mathcal{O}'_2$. Nonetheless, our evaluation suggests that the number of remaining violations after repair is typically small (See Section 5 in [69], Appendix F).

5.5 Ontology Approximation

The Optique platform supports ontologies expressible in the OWL 2 QL profile of the OWL 2 ontology language, which was specifically designed for efficient data access. Imported or layered, OWL 2 ontologies that cannot be captured in OWL 2 QL are automatically approximated into OWL 2 QL.

In this section we first formally provide a general, parametric, and semantically well-founded definition of maximal sound approximation of a DL ontology. Our semantic definition captures and generalizes previous approaches to ontology approximation [13, 19, 51, 56]. In particular, our approach builds on the preliminary work presented in [19], which proposed a similar, although non-parameterized, notion of maximal sound approximation.

Then,we present an algorithm for computing maximal sound approximation according to the above parametric semantics, when the source ontology language is OWL 2 and the target ontology language is OWL 2 QL. In particular, we focus on the *local semantic approximation (LSA)* and the *global semantic approximation (GSA)* of a source ontology. These two notions of approximation correspond to the cases when the parameter of our semantics is, respectively, minimal and maximal. Informally, the LSA of an ontology is obtained by considering (and reasoning over) one axiom α of the source ontology at a time, so this technique tries to approximate α independently of the rest of the source ontology. On the contrary, the GSA tries to approximate the source ontology by considering all its axioms (and reasoning over such axioms) at the same time. As a consequence, the GSA is potentially able to approximate "better" than the LSA, while the LSA appears in principle computationally less expensive than the GSA. Notably, in the case of OWL 2 QL, the result of approximating an ontology according to GSA is logically equivalent to the one obtained according to the notion of approximation given in [56], which has been shown to be very well-suited for query answering purposes.

An empirical evaluation of our methods can be found in the Appendix J.

5.5.1 Global semantic approximation

In what follows, we illustrate our notion of approximation in a target language $\mathcal{L}_T$ of an ontology $\mathcal{O}_S$ in a language $\mathcal{L}_S$.

Typically, when discussing approximation, one of the desirable properties is that of soundness. Roughly speaking, when the object of approximation is a set of models, this property requires that the set of models of the approximation is a superset of those of the original ontology. Another coveted characteristic of the computed ontology is that it be the "best" approximation of the original ontology. In other words, the need of keeping a minimal distance between the original ontology and the ontology resulting from its approximation is commonly perceived.

On the basis of these observations, the following definition of approximation in a target language $\mathcal{L}_T$ of a satisfiable $\mathcal{L}_S$-ontology is very natural.

Definition 5.5.1 *Let $\mathcal{O}_S$ be a satisfiable $\mathcal{L}_S$-ontology, and let $\Sigma_{\mathcal{O}_S}$ be the set of predicate and constant symbols occurring in $\mathcal{O}_S$. An $\mathcal{L}_T$-ontology $\mathcal{O}_T$ over $\Sigma_{\mathcal{O}_S}$ is a* global semantic approximation *(GSA) in $\mathcal{L}_T$ of $\mathcal{O}_S$ if both the following statements hold:*

(i) $Mod(\mathcal{O}_S) \subseteq Mod(\mathcal{O}_T)$;

(ii) there is no $\mathcal{L}_T$-ontology $\mathcal{O}'$ over $\Sigma_{\mathcal{O}_S}$ such that $Mod(\mathcal{O}_S) \subseteq Mod(\mathcal{O}') \subset Mod(\mathcal{O}_T)$.

We denote with $globalApx(\mathcal{O}_S, \mathcal{L}_T)$ the set of all the GSAs in $\mathcal{L}_T$ of $\mathcal{O}_S$.

In the above definition, statement (i) imposes the soundness of the approximation, while statement (ii) imposes the condition of "closeness" in the choice of the approximation.

We observe that an $\mathcal{L}_T$-ontology which is the GSA in $\mathcal{L}_T$ of $\mathcal{O}_S$ may not exist. This is the case when, for each $\mathcal{L}_T$ ontology $\mathcal{O}'_T$ satisfying statement (i) of Definition 5.5.1, there always exists an $\mathcal{L}_T$-ontology $\mathcal{O}''_T$ which satisfies statement (i), but for which we have that $Mod(\mathcal{O}_S) \subseteq Mod(\mathcal{O}''_T) \subset Mod(\mathcal{O}'_T)$.

As shown in the paper in Appendix H a sufficient condition for the existence of the GSA in a language $\mathcal{L}_T$ of an ontology $\mathcal{O}_S$ is that the set of non-equivalent axioms in $Axioms(\mathcal{L}_T)$ that one can generate over Σ is finite. In the paper it is also shown that if $\mathcal{L}_T$ is a closed language, then for each $\mathcal{O}'$ and $\mathcal{O}''$ belonging to *globalApx*$(\mathcal{O}_S, \mathcal{L}_T)$, we have that $\mathcal{O}'$ and $\mathcal{O}''$ are logically equivalent. In other words, if the target language is closed, then, up to logical equivalence, the GSA is unique.

The notion of *entailment set* [56], which we introduce below, allows us to provide more constructive conditions, equivalent to those in Definition 5.5.1.

Definition 5.5.2 *Let $\Sigma_{\mathcal{O}}$ be the set of predicate and constant symbols occurring in $\mathcal{O}$, and let $\mathcal{L}'$ be a language. The* entailment set *of $\mathcal{O}$ with respect to $\mathcal{L}'$, denoted as $\mathsf{ES}(\mathcal{O}, \mathcal{L}')$, is the set of axioms from $Axioms(\mathcal{L}')$ that only contain predicates and constant symbols from $\Sigma_{\mathcal{O}}$ and that are entailed by $\mathcal{O}$.*

In other words, we say that an axiom α belongs to the entailment set of an ontology $\mathcal{O}$ with respect to a language $\mathcal{L}'$, if α is an axiom in $Axioms(\mathcal{L}')$ built over the signature of $\mathcal{O}$ and for each interpretation $\mathcal{I} \in Mod(\mathcal{O})$ we have that $\mathcal{I} \models \alpha$.

Clearly, given an ontology $\mathcal{O}$ and a language $\mathcal{L}'$, the entailment set of $\mathcal{O}$ with respect to $\mathcal{L}'$ is unique.

We have that, given a satisfiable $\mathcal{L}_S$-ontology $\mathcal{O}_S$ and a satisfiable $\mathcal{L}_T$-ontology $\mathcal{O}_T$, the following conditions hold:

(a) $Mod(\mathcal{O}_S) \subseteq Mod(\mathcal{O}_T)$ if and only if $\mathsf{ES}(\mathcal{O}_T, \mathcal{L}_T) \subseteq \mathsf{ES}(\mathcal{O}_S, \mathcal{L}_T)$;

(b) there is no $\mathcal{L}_T$-ontology $\mathcal{O}'$ such that $Mod(\mathcal{O}_S) \subseteq Mod(\mathcal{O}') \subset Mod(\mathcal{O}_T)$ if and only if there is no $\mathcal{L}_T$-ontology $\mathcal{O}''$ such that $\mathsf{ES}(\mathcal{O}_T, \mathcal{L}_T) \subset \mathsf{ES}(\mathcal{O}'', \mathcal{L}_T) \subseteq \mathsf{ES}(\mathcal{O}_S, \mathcal{L}_T)$.

Therefore, every ontology $\mathcal{O}_T$ which is a GSA in $\mathcal{L}_T$ of an ontology $\mathcal{O}_S$ is also an approximation in $\mathcal{L}_T$ of $\mathcal{O}_S$ according to the paper in Appendix I, and, for some languages, this also corresponds to the approximation in [56].

5.5.2 K-approximation

The computation of a GSA can be a very challenging task even when approximating into tractable fragments of OWL 2 [54] because it is necessary to reason over the ontology as a whole. Consequently, we introduce a new notion of approximation, in which we do not reason over the entire ontology but only over portions of it. At the basis of this new notion, which we call *k-approximation*, is the idea of obtaining an approximation of the original ontology by computing the global semantic approximation of each set of k axioms of the original ontology in isolation. Below we give a formal definition of the k-approximation.

In what follows, given an ontology $\mathcal{O}$ and a positive integer k such that $k \leq |\mathcal{O}|$, we denote with $subset_k(\mathcal{O})$ the set of all the sets of cardinality k of axioms of $\mathcal{O}$.

Definition 5.5.3 *Let $\mathcal{O}_S$ be a satisfiable $\mathcal{L}_S$-ontology and let $\Sigma_{\mathcal{O}_S}$ be the set of predicate and constant symbols occurring in $\mathcal{O}_S$. Let $\mathcal{U}_k = \{\mathcal{O}_i^j \mid \mathcal{O}_i^j \in globalApx(\mathcal{O}_i, \mathcal{L}_T)$, such that $\mathcal{O}_i \in subset_k(\mathcal{O}_S)\}$. An $\mathcal{L}_T$-ontology $\mathcal{O}_T$ over $\Sigma_{\mathcal{O}_S}$ is a* k-approximation *in $\mathcal{L}_T$ of $\mathcal{O}_S$ if both the following statements hold:*

- $\bigcap_{\mathcal{O}_i^j \in \mathcal{U}_k} Mod(\mathcal{O}_i^j) \subseteq Mod(\mathcal{O}_T)$;
- *there is no $\mathcal{L}_T$-ontology $\mathcal{O}'$ over $\Sigma_{\mathcal{O}_S}$ such that* $\bigcap_{\mathcal{O}_i^j \in \mathcal{U}_k} Mod(\mathcal{O}_i^j) \subseteq Mod(\mathcal{O}') \subset Mod(\mathcal{O}_T)$.

Once again, using the notion of entailment set, we can give a constructive condition for the k-approximation. Indeed, given a satisfiable $\mathcal{L}_S$-ontology $\mathcal{O}_S$ and a satisfiable $\mathcal{L}_T$-ontology $\mathcal{O}_T$, both over $\Sigma_{\mathcal{O}_S}$, we have that $\mathcal{O}_T$ is a k-approximation in $\mathcal{L}_T$ of $\mathcal{O}_S$ if and only if:

(i) $\mathsf{ES}(\mathcal{O}_T, \mathcal{L}_T) \subseteq \mathsf{ES}(\bigcup_{\mathcal{O}_i \in subset_k(\mathcal{O}_S)} \mathsf{ES}(\mathcal{O}_i, \mathcal{L}_T), \mathcal{L}_T)$;

(ii) there is no $\mathcal{L}_T$-ontology $\mathcal{O}'$ over $\Sigma_{\mathcal{O}_S}$ such that:

$$\mathsf{ES}(\mathcal{O}_T, \mathcal{L}_T) \subset \mathsf{ES}(\mathcal{O}', \mathcal{L}_T) \subseteq \mathsf{ES}(\bigcup_{\mathcal{O}_i \in subset_k(\mathcal{O}_S)} \mathsf{ES}(\mathcal{O}_i, \mathcal{L}_T), \mathcal{L}_T).$$

Note that if $k = |\mathcal{O}_S|$, the k-approximation actually coincides with the GSA. At the other end of the spectrum, we have the case in which $k = 1$. Here we are treating each axiom α in the original ontology in isolation, i.e., we are considering ontologies formed by a single axiom α. We refer to this approximation as *local semantic approximation* (LSA).

Example 5.5.1 *Consider the following OWL 2 ontology $\mathcal{O}$.*

$$\begin{array}{lllll} \mathcal{O} & = & \{ & A \sqsubseteq B \sqcup C & B \sqsubseteq D & A \sqsubseteq \exists R.D \\ & & & B \sqcap C \sqsubseteq F & C \sqsubseteq D & \exists R.D \sqsubseteq E \ \}. \end{array}$$

The following ontology is a GSA in OWL 2 QL of $\mathcal{O}$.

$$\mathcal{O}_{GSA} = \{ \begin{array}{llll} A \sqsubseteq D & B \sqsubseteq D & A \sqsubseteq \exists R & A \sqsubseteq \exists R.D \\ A \sqsubseteq E & C \sqsubseteq D & D \sqsubseteq F & \}. \end{array}$$

Indeed, it is possible to show that each axiom entailed by $\mathcal{O}_{GSA}$ is also entailed by $\mathcal{O}$, and that it is impossible to build an OWL 2 QL ontology $\mathcal{O}'$ such that $\mathsf{ES}(\mathcal{O}_{GSA}, OWL\ 2\ QL) \subset \mathsf{ES}(\mathcal{O}', OWL\ 2\ QL) \subseteq \mathsf{ES}(\mathcal{O}, OWL\ 2\ QL)$.

Computing the LSA in OWL 2 QL of $\mathcal{O}$, i.e., its 1-approximation in OWL 2 QL, we obtain the following ontology.

$$\mathcal{O}_{LSA} = \{ \begin{array}{lll} B \sqsubseteq D & A \sqsubseteq \exists R & \\ C \sqsubseteq D & A \sqsubseteq \exists R.D & \}. \end{array}$$

□

The example shows that $Mod(\mathcal{O}) \subset Mod(\mathcal{O}_{GSA}) \subset Mod(\mathcal{O}_{LSA})$, which means that the ontology $\mathcal{O}_{GSA}$ approximates $\mathcal{O}$ better than $\mathcal{O}_{LSA}$. This expected result is a consequence of the fact that reasoning over each single axiom in $\mathcal{O}$ in isolation does not allow for the extraction all the OWL 2 QL consequences of $\mathcal{O}$.

5.5.3 Approximation in OWL 2 QL

The computation of the approximation as defined above can be a very challenging task even when approximating into a lower-complexity description logics such as OWL 2 QL or the languages of the *DL-Lite* family. For an in-depth investigation of the problem of approximating ontologies in languages belonging to this family of description logics, we refer the reader to the paper in the Appendix I. Instead we now focus on the problem of approximating ontologies in OWL 2 with ontologies in OWL 2 QL.

In this scenario, some of the issues mentioned in the paper in Appendix I no longer present themselves. Specifically, the result of the approximation in OWL 2 QL of an OWL 2 ontology is unique, in the sense that if two ontologies $\mathcal{O}'$ and $\mathcal{O}''$ are both approximations in OWL 2 QL of the same OWL 2 ontology $\mathcal{O}$, then they are logically equivalent. This is a consequence of the fact that OWL 2 QL is a closed language. Moreover, since the set of non-equivalent OWL 2 QL axioms that one can generate over a signature Σ is finite, the k-approximation in OWL 2 QL of an OWL 2 ontology always exists.

As stated in Theorem 3 of the paper in Appendix H, we have that the k-approximation in OWL 2 QL of an OWL 2 ontology $\mathcal{O}_S$ coincides with the set $\bigcup_{\mathcal{O}_i \in subset_k(\mathcal{O}_S)} \mathsf{ES}(\mathcal{O}_i, OWL\ 2\ QL)$, i.e., the union of the entailment sets of each subset of cardinality k of $\mathcal{O}_S$ with respect to OWL 2 QL.

Algorithm 2: $computeKApx(\mathcal{O}, k)$

```
Input: a satisfiable OWL 2 ontology $\mathcal{O}$, a positive integer $k$ such that $k \leq |\mathcal{O}|$
Output: an OWL 2 QL ontology $\mathcal{O}_{Apx}$
begin
    $\mathcal{O}_{Apx} \leftarrow \emptyset$;
    foreach ontology $\mathcal{O}_i \in subset_k(\mathcal{O}_S)$
        $\mathcal{O}_{Apx} \leftarrow \mathcal{O}_{Apx} \cup \mathsf{ES}(\mathcal{O}_i, OWL\ 2\ QL)$;
    return $\mathcal{O}_{Apx}$;
end
```

Notably, we observe that for $k = |\mathcal{O}_S|$ the k-approximation $\mathcal{O}_T$ in OWL 2 QL of $\mathcal{O}_S$ coincides with its entailment set in OWL 2 QL. This means that $\mathcal{O}_T$ is also the approximation in OWL 2 QL of $\mathcal{O}_S$ according to the notion of approximation presented in [56]. Therefore, all the properties that hold for the semantics in [56] also hold for the GSA. In particular, the evaluation of a conjunctive query q without non-distinguished variables over $\mathcal{O}_S$ coincides with the evaluation of q over $\mathcal{O}_T$ (Theorem 5 in [56]).

In the paper in Appendix H Algorithm 2, for computing the k-approximation of an $\mathcal{L}_S$-ontology $\mathcal{O}_S$ in OWL 2 QL, is given.

The *computeKApx* algorithm first computes every subset with size k of the original ontology $\mathcal{O}_S$. Then, it computes the ontology which is the result of the k-approximation in OWL 2 QL of the ontology in input as the union of the entailment sets with respect to OWL 2 QL of each such subset. It is clear that the key point of the algorithm is the computation of the entailment set.

5.5.4 Computing the entailment set in OWL 2 QL

The computation of $\mathsf{ES}(\mathcal{O}, OWL\ 2\ QL)$ is in general very costly, as highlighted also in [13] and [56], since it requires the invocation of reasoning services over an OWL 2 ontology $\mathcal{O}$ for checking, for every assertion $\alpha \in \mathsf{ES}(\mathcal{O}, OWL\ 2\ QL)$, if $\mathcal{O} \models \alpha$. This task is performed by invoking an OWL 2 oracle which can be implemented by an OWL 2 reasoner.

A naive algorithm for computing the entailment set with respect to OWL 2 QL can be easily obtained from the one given in [56] for *DL-Lite* languages. We can summarize it as follows. Let $\mathcal{O}$ be an ontology and let $\Sigma_{\mathcal{O}}$ be the set of predicate and constant symbols occurring in $\mathcal{O}$. The algorithm first computes the set Γ of axioms in $Axioms(OWL\ 2\ QL)$ which can be built over $\Sigma_{\mathcal{O}}$, and then, for each axiom $\alpha \in \Gamma$ such that $\mathcal{O} \models \alpha$, adds α to the set $\mathsf{ES}(\mathcal{O}, OWL\ 2\ QL)$. In practice, to check if $\mathcal{O} \models \alpha$ one can use an OWL 2 reasoner. Since each invocation of the OWL 2 reasoner is N2ExpTime, the computation of the entailment set can be very costly [13].

A more efficient technique for its computation is the one presented in the paper in Appendix I. The key idea of this technique is to limit the number of invocations to the OWL 2 reasoner, by exploiting the knowledge acquired through a preliminary exploration of the ontology. This optimization technique is presented in Section 5 of the paper. To understand the basic idea behind this technique, consider, for example, an ontology $\mathcal{O}$ that entails the inclusions $A_1 \sqsubseteq A_2$ and $P_1 \sqsubseteq P_2$, where A_1 and A_2 are concepts and P_1 and P_2 are roles. Exploiting these inclusions we can deduce the hierarchical structure involving general concepts that can be built on these four predicates. For instance, we know that $\exists P_2.A_2 \sqsubseteq \exists P_2$, that $\exists P_2.A_1 \sqsubseteq \exists P_2.A_2$, that $\exists P_1.A_1 \sqsubseteq \exists P_2.A_1$, and so on. We begin by invoking the OWL 2 oracle, asking for the children of the general concepts which are in the highest position in the hierarchy. So, first compute the subsumees of $\exists P_2$ through the OWL 2 reasoner. If there are none, we avoid invoking the oracle asking for the subsumees of $\exists P_2.A_2$ and so on.

5.6 Provenance in Bootstrapped Mappings

Data provenance has been identified as one of the requirements in the Statoil use case (see deliverable D9.1) since some of the queries require the comparison of data coming from different databases (see deliverable D9.2).

The O&M boostrapper can automatically attach provenance information to the direct mappings without extra cost. Thus, this information can be exploited to answer queries involving provenance information (e.g. comparing results from different databases).

In this section we present three different levels of granularity at which we can provide provenance annotations. For a particular application, the expert responsible of modeling the domain can choose any of these levels of granularity or a combination of them depending on the competency questions [73] that need to be addressed by the OBDA system and the capabilities of the underlying system. We have also extended PROV-O [8] with additional predicates for convenience in the OBDA domain, we will refer to these predicates with the "`obdaprov:`" prefix.

5.6.1 Provenance model

The PROV-O model focuses on three main concepts: `prov:Entity`, `prov:Agent` and `prov:Activity`. The examples in the previous section show the focus on the relation between the entities. The agents and activities could be useful in some particular cases, but in general the agent will be the specific OBDA system (a `prov:SoftwareAgent`) and the activity will be OBDA query answering or triple materialization, a form of creation (`prov:Create`). Normally in a OBDA scenario the main concern will be the relation between the entities, i.e. the original information that is accessed and the information that is derived from it (`prov:wasDerivedFrom`). Therefore the extension of the PROV-O model focuses on entities.

In particular, we extend the model in PROV-O to consider the elements that are particular to a OBDA setting, where `OBDADatabase`, `OBDATable`, `OBDAColumn` and `OBDARow` are subclasses both of `prov:Entity` and the corresponding elements in a regular database. These entities relate to each other, allowing to locate a particular row or column in a table and a table in a database. We only need to state the specific location of some `:OBDAEntity` by specifying a set of properties, like `hasDatabase`, `hasTable` or `hasColumn`.

5.6.2 Provenance at URI level

URIs identify the smallest fragments of information produced by OBDA systems. To provide the provenance information we can simply state that the provenance of some URI corresponds to some `OBDAEntity`. A mapping to add this kind of assertions would look like this:

```
:TriplesMap_prov1
  a rr:TriplesMap ;
  rr:logicalTable [ rr:tableName "Field" ] ;
  rr:subjectMap [
    rr:template "http://base_uri/Field/{id}" ;
    rr:class prov:Entity
  ] ;
  rr:predicateObjectMap [
    rr:predicate prov:wasDerivedFrom ;
    rr:objectMap [
      rr:template "http://base_uri/OBDAURI/URI_field_{id}"
    ]
  ] .
```

With this mapping we add a new entity that is the source of the generated URIs, and we can generate all the provenance information for this entity with a new mapping:

```
:TriplesMap_prov2
  a rr:TriplesMap ;
  rr:logicalTable [ rr:tableName "Field" ] ;
  rr:subjectMap [
    rr:template "http://base_uri/OBDAURI/URI_field_{id}" ;
    rr:class prov:Entity
  ] ;
  rr:predicateObjectMap [
    rr:predicate obdaprov:hasDatabase ;
    rr:objectMap [ rr:template "Database_Running_Example" ]
  ] ;
  rr:predicateObjectMap [
    rr:predicate obdaprov:hasTable ;
    rr:objectMap [ rr:template "Field" ]
  ] ;
  rr:predicateObjectMap [
    rr:predicate obdaprov:hasColumn ;
    rr:objectMap [ rr:template "id" ]
  ] .
```

It may be worthy to point out that several mappings can be responsible of the generation of the same URIs. This is specially important if some join has to be performed between several data sources. These URIs are derived from several different entities, and the provenance will correspond to the union of the sources responsible for the generation of these URIs.

Additionally, note that `rr:template "http://base_uri/OBDAURI/URI_field_id"` in `:TriplesMap_prov1` could have been replaced with a `rr:parentTriplesMap` to `:TriplesMap_prov2`.

5.6.3 Provenance at triple level

To include the information about provenance in the mapping assertion we can include this information as constants in the head of the mapping. This can be done either manually or automatically if there is some automatic process for the generation of the mappings. By using reification to keep track of this metainformation we can obtain the following result for the first `rr:predicateObjectMap` in the `:TriplesMap1` in Section 5.2:

```
:TriplesMap_prov3
  a rr:TriplesMap ;
  rr:logicalTable [ rr:tableName "Field" ] ;
  rr:subjectMap [
    rr:template "http://base_uri/statement/{id}" ;
    rr:class rdf:Statement ;
    rr:class prov:Entity
  ] ;
  rr:predicateObjectMap [
    rr:predicate rdf:subject ;
    rr:objectMap [
      rr:template "http://base_uri/Field/{id}"
    ]
  ] ;
  rr:predicateObjectMap [
    rr:predicate rdf:predicate ;
    rr:object <http://base_uri/name>
  ] ;
  rr:predicateObjectMap [
    rr:predicate rdf:object ;
    rr:objectMap [ rr:column "name" ]
  ] ;
  rr:predicateObjectMap [
    rr:predicate obdaprov:wasDerivedWithOBDAMapping ;
    rr:object :TriplesMap1
  ] ;
  rr:predicateObjectMap [
    rr:predicate prov:wasDerivedFrom ;
    rr:object [
      rr:template "http://base_uri/OBDAEntity/URI_field_name_{id}"
    ]
  ] .
```

A new mapping (analogous to `:TriplesMap_prov2` in the previous section) would be needed to specify the characteristics of `"http://base_uri/OBDAEntity/URI_field_name_id"`, i.e. its database, table and column. All sorts of provenance information can be added to the triple reified in the previous map, for example the `prov:Activity` activity that generates it (`:ourOBDAActivity`) or the system (`:ourOBDASystem`) that as a `prov:SoftwareAgent` performs this activity. This depends on the competency questions to be addressed for each particular use case.

As a limitation, the mappings would generate additional triples due to reification.

5.6.4 Provenance at graph level

For many applications the triple granularity will not be needed. If a set of triples share some details about their provenance (e.g. the source database is the same) then a more efficient solution can be annotating provenance for all of them together in a RDF named graph. This can easily be done by modifying the original mappings and specifying a graph containing the triples that share some provenance detail and creating additional mappings to state the provenance of that graph. The statements about the provenance can be included in the same graph, allowing for a closed recursion.

If graphs refer to greater entities (e.g. data sources) their specification will very probably be more stable. Adding a new graph with details about provenance may be an usual operation that can be performed manually (or semi-automatically) when adding a new data source to the system. Therefore, the information about the provenance of the graph can be stated statically without requiring additional mappings for this, for example:

```
<http://base_uri/graph/Field>
```

```
 a Entity ;
 prov:wasDerivedFrom "http://base_uri/OBDAEntity/Database_Running_Example" .
```

As in previous examples, the properties of `"http://base_uri/OBDAEntity/table_Field"` would be defined in a new map specifying the original database and any other details to consider.

Chapter 6

Post-Bootstrapping Analysis

The previous section has described automatic techniques to bootstrap an ontology and a set of mapping given a database schema as input. Additionally, the bootstrapped ontology can be automatically aligned to a domain ontology and/or the boostrapped mapping can be linked to a domain ontology. Finally, the aligned ontology can be approximated if it is outside the OWL 2 QL profile.

The next three subsections present complementary (semi-automatic) efforts within Optique to extend and validate the bootstrapped ontology and mappings. These efforts will be further developed within the Optique's O&M analysis module. Section 6.1 presents a semi-automatic method to create a set of mappings given an ontology and a database schema as inputs. In Section 6.2 we present how the bootstrapped ontology and mappings can be validated and/or edited. Finally, Section 6.3 provides a methodology for the manual construction of the ontology and the mappings.

6.1 Semi-Automatic Ontology Layering

In addition to the bootstrapping module, the O&M component also contains an *Ontology Layering Module* that offers to layer an input ontology over an input DB schema resulting in a set of direct mappings between the ontology and the schema. Essentially, a user constructs mappings based on automatically generated *suggested candidates*, thus operating semi-automatically. Ontology layering relies on the IncMap system [58] (see Appendix B), which is provided as a special module inside the platform. Internally, IncMap represents both the ontology and schema uniformly, using a structure-preserving meta-graph structure called *IncGraph*. Thus, correspondences can be calculated and visualized directly between the two data models. The module then computes ranked correspondences between elements of the graphs using lexical and structural similarities, based on the Similarity Flooding algorithm of Melnik et al. [53], and converts the correspondences into direct mappings between the ontology and schema. Human verification is provided through the semi-automatic nature of the layering process, where each suggested mapping needs to be accepted by an expert user before it becomes effective. IncMap operates incrementally, i.e., it uses human input to rerank correspondences after each round of interaction. Verifications performed by a user (i.e., verified mappings) can thus be used to improve the quality of subsequent suggestions in the structural neighborhood. As a major difference to bootstrapping and alignment, layering can map user-specified fragments of DB schemata to user-specified fragments of ontologies.

6.2 Validation of the boostrapped ontology and mappings

The axioms computed by the ontology bootstrapper (including the ontology-to-ontology alignments) are presented to the user for verification, i.e., the user can edit, accept, or discard candidate axioms.[1]

For use cases requiring very accurate alignments, our ontology alignment system LogMap also supports user interaction during the alignment process in order to keep the number of wrong correspondences to a minimum.

Additionally, the R2RML Mapping Editor of the Optique platform is tailored towards W3C R2RML mappings for which direct mappings is a special case, and was evaluated with encouraging results [57]. Manual edition of R2RML mappings is a time consuming and error prone process. To ease this issue, the R2RML editor provides

[1]Note that, to the time being the platform has limited support for in-line ontology editing and visualization. However, ontologies from the platform can easily be exported to matured, third party tools such as Protege (`http://protege.stanford.edu/`). Additionally the platform also provides functionality to import ontologies (see Deliverable D2.4 for details).

an intuitive mapping visualisation, semi-automatic suggestions of mapping corrections, and step-by-step wizards for writing complex (non direct) mappings.

6.3 Guidelines for the manual construction of an OBDA specification

“Guessing” an ontology from a database schema is no easy task, since the database modelling step that produces a database schema from (explicit or implicit) knowledge of a domain is typically lossy. Automatic bootstrapping means to make a best effort at reverse engineering this step and is necessarily imperfect. However, depending on the quality of the data source schemata, the results often provide a very good starting point for later manual optimisations that can be applied as required. Hence, in Optique we have defined a methodology for the development and maintenance of an OBDA specification based on the functionalities offered by the Optique system.

Towards this goal, we have provided, as a starting point, a description of some initial guidelines for “manually” building an OBDA specification.

This is already a significant contribution, since the literature in the field does not actually present any methodology for the specification of a full-fledged OBDA system. Moreover, our previous experience in the field, as well as our interactions with the project use cases, clearly showed that such a methodology is actually far from being trivial. In particular, while there exist several approaches and methodologies for ontology development and ontology engineering (e.g., [26], [33], [11], [34]), research in data integration and data exchange has only very partially dealt with the issue of mapping development and maintenance (e.g., [46], [27], [36], [47]). Therefore, we miss a real methodology for OBDA specification, that is, an approach that brings together ontology development, data source analysis, and mapping development.

The document reporting such initial guidelines is reported in Appendix C. In the next years, we intend to turn the above initial guidelines into a true methodology for OBDA specification, facing the issue of combining ontology development, data source analysis, and mapping development in a uniform and comprehensive approach, as well as taking into account the functionalities that will be provided by the Optique platform.

Chapter 7

Integration with the Optique Platform

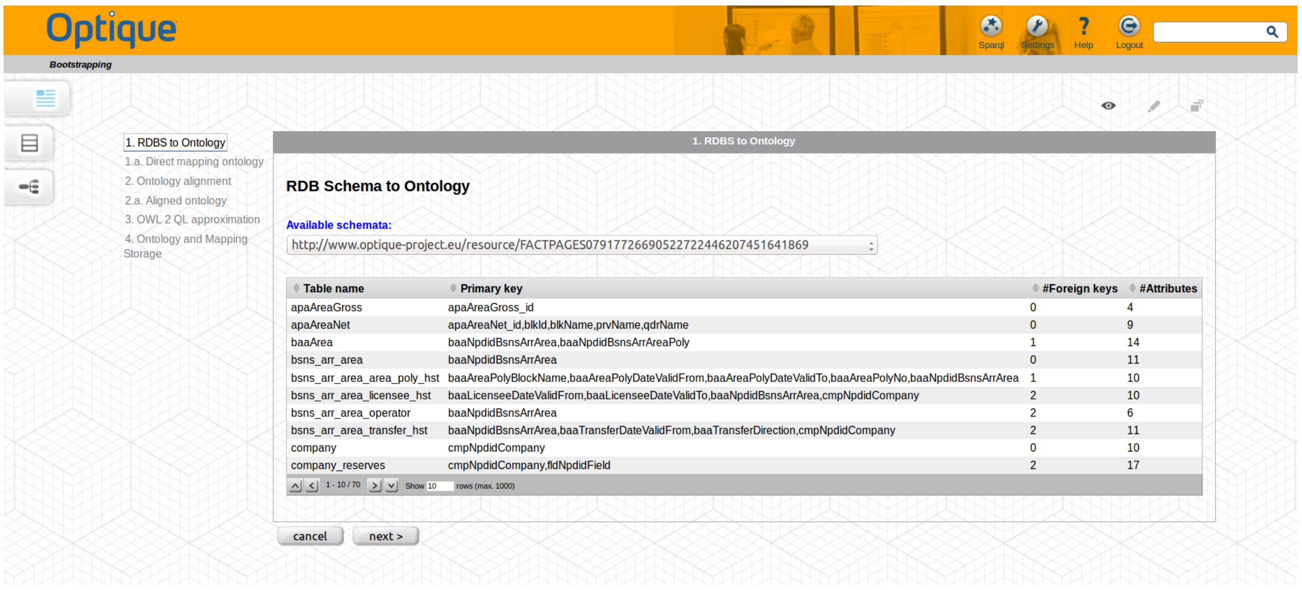
The Optique O&M component is equipped with an O&M bootstrapper which integrated the techniques described in Section 5.

The O&M boostrapper is fully compliant with the Optique platform APIs: the ontology API, the relational metadata API, the R2RML mapping management API. See Deliverables D2.3 (*First Prototype of the Core Platform*) and D2.4 (*Second Prototype of the Core Platform*) for details about these APIs.

The O&M boostrapper component retrieves the required metadata from the platfrorms' shared metadata repository using the Relational Metadata API. The O&M boostrapper also integrates the ontology matching system described in Section 5.4 and an ontology approximation module to transform the resulting ontology if it is outside the desired OWL 2 profile (see Section 5.5). The O&M bootstrapper uses the Ontology API and the Mapping Management API to store the resulting assests (i.e. ontology and mappings) back to the platform.

A *demo* paper describing the first version of the Optique system, including the O&M bootstrapper, was accepted in the 2013 edition of the International Semantic Web Conference [43] (see Appendix K). Futhermore, we recently submitted a demo paper to an international conference (see Appendix L). The O&M bootstrapping is accessible from the Optique's demo web page: `http://fact-pages.fluidops.net/` Demonstration videos are also available at the following address: `https://www.youtube.com/user/optiqueproject`.

Next sections summarizes the different steps of the O&M boostrapper.



RDB Schema to Ontology

Available schemata:

http://www.optique-project.eu/resource/FACTPAGES07917726690522722446207451641869

Table name	Primary key	#Foreign keys	#Attributes
apaAreaGross	apaAreaGross_id	0	4
apaAreaNet	apaAreaNet_id,blkId,blkName,prvName,qdrName	0	9
baaArea	baaNpdidBsnsArrArea,baaNpdidBsnsArrAreaPoly	1	14
bsns_arr_area	baaNpdidBsnsArrArea	0	11
bsns_arr_area_area_poly_hst	baaAreaPolyBlockName,baaAreaPolyDateValidFrom,baaAreaPolyDateValidTo,baaAreaPolyNo,baaNpdidBsnsArrArea	1	10
bsns_arr_area_licensee_hst	baaLicenseeDateValidFrom,baaLicenseeDateValidTo,baaNpdidBsnsArrArea,cmpNpdidCompany	2	10
bsns_arr_area_operator	baaNpdidBsnsArrArea	2	6
bsns_arr_area_transfer_hst	baaNpdidBsnsArrArea,baaTransferDateValidFrom,baaTransferDirection,cmpNpdidCompany	2	11
company	cmpNpdidCompany	0	10
company_reserves	cmpNpdidCompany,fldNpdidField	2	17

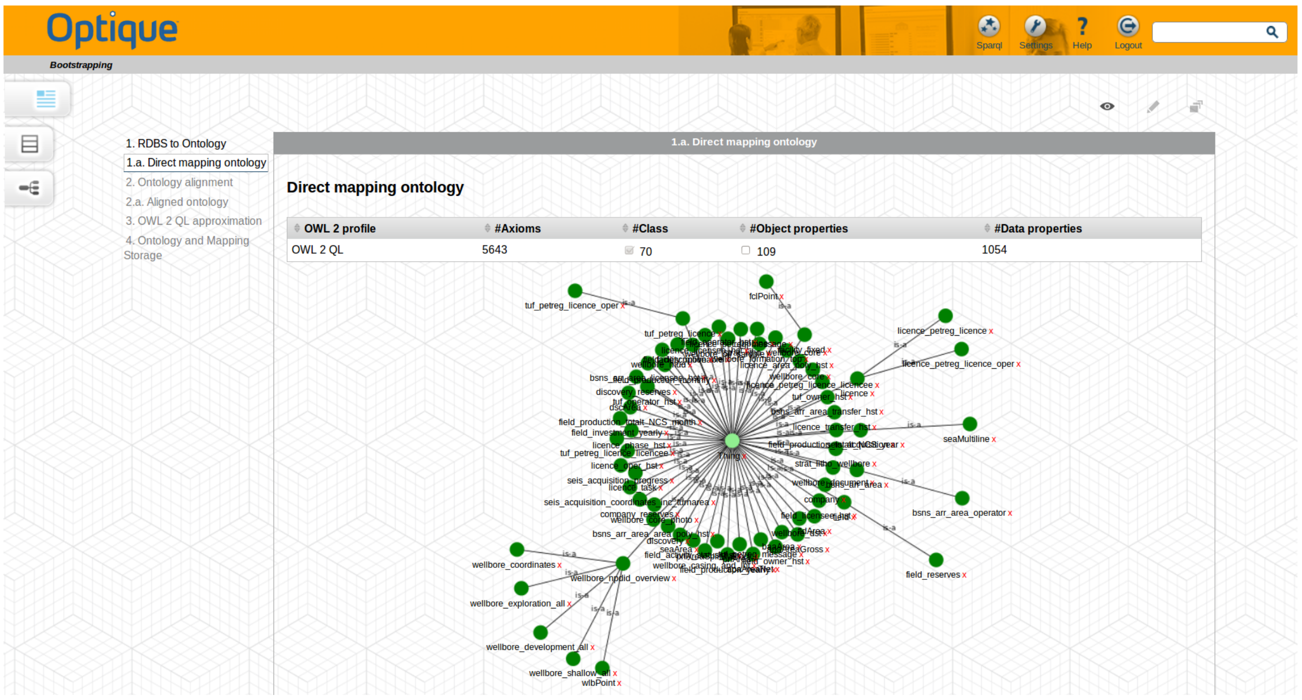
1 - 10 / 70 Show 10 rows (max. 1000)

Figure 7.1: O&M bootstrapper: selection of the RDB Schema

7.1 RDB Schema Selection and Bootstrapping

Once the platform has been connected successfully to a relational data base (RDB) (see Section 3.2.1 of Deliverable D2.4), the O&M can be used. Figure 7.1 shows the first step of the O&M bootstrapper. The available RDB schemas are listed and, upon selection, the metadata of the desired RDB schema is visualized. Once the "next" button is pressed the O&M bootstrapper, following the specification and semantics described in Sections 5.2 and 5.3 creates the bootstrapped mappings and ontology.

Figure 7.2 shows the second step of the O&M bootstrapper where an overview of the boostrapped (direct mapping) ontology is visualized. Apart from ontology classes, the interface also allows the (optional) visualization of object properties.



Direct mapping ontology

OWL 2 profile	#Axioms	#Class	#Object properties	#Data properties
OWL 2 QL	5643	70	109	1054

Figure 7.2: O&M bootstrapper: bootstrapped (direct mapping) ontology visualization

7.1.1 Ontology Alignment

As described in Section 5.4, the O&M bootstrapper integrates the ontology alignment system LogMap. LogMap identifies correspondences or alignments between the bootstrapped ontology and a given domain ontology.

Figure 7.3 shows the alignment of the boostrapped ontology (green nodes) with a domain ontology (blue nodes). The dark-red links represent the alignments extracted by LogMap.

7.1.2 Ontology approximation

The O&M bootstrapper uses the ontology approximation module described in Section 5.5 to approximate the resulting ontology if it is outside the desired OWL 2 QL profile (i.e. current language supported by the Optique OBDA system). Figure 7.4 shows the O&M bootstrapper interface for the ontology approximation.

The implemented approximation module, other than computing the approximation in OWL 2 QL of the ontology, also returns a summary *log* specifying how the module handled each axiom of he original ontology. After the approximation, in the report each axiom of the original ontology is categorized into one of three categories, based on the type of approximation it has undergone:

- *Unapproximable axioms*: all the semantics of the axiom is lost, which means that given an OWL 2 axiom α of the original ontology, the entailment set of α in OWL 2 QL is empty.

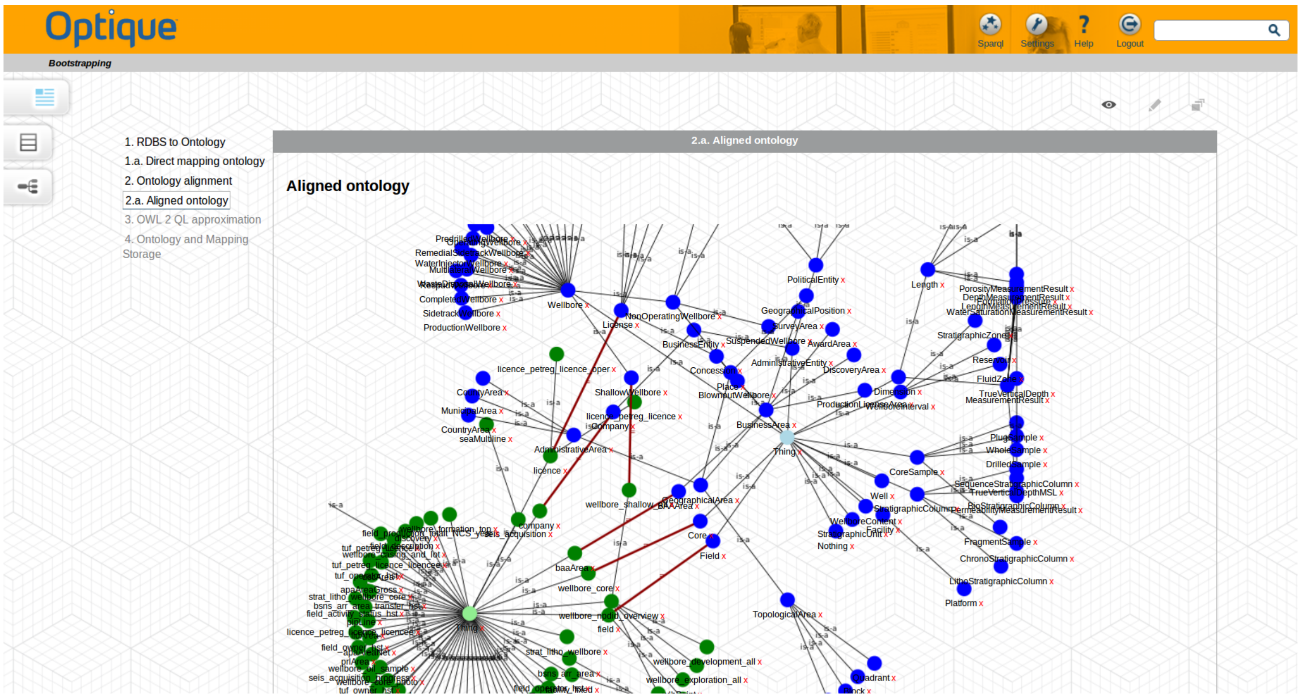
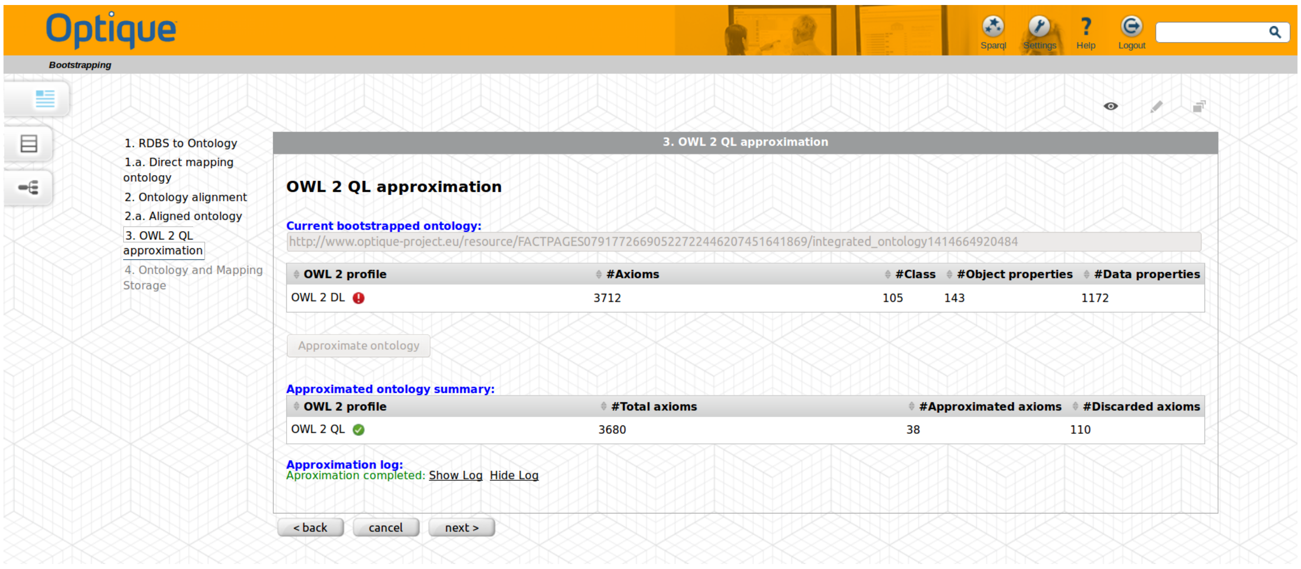


Figure 7.3: O&M bootstrapper: alignment of the DMO ontology with a SOTA ontology

- *Equivalently rewritten axioms*: all the semantics of the axiom is preserved, which means that given an OWL 2 axiom α of the original ontology, it is rewritten by the approximation module as a set of OWL 2 QL axioms $\mathcal{S} \subseteq \mathsf{ES}(\alpha, OWL\ 2\ QL)$ such that $\mathcal{S} \models \alpha$.
- *Approximated axioms*: only a portion of the semantics of the axiom is preserved, which means that given an OWL 2 axiom α of the original ontology, it is rewritten by the approximation module as a set of OWL 2 QL axioms $\mathcal{S} \subseteq \mathsf{ES}(\alpha, OWL\ 2\ QL)$ such that $\mathcal{S} \not\models \alpha$.



OWL 2 QL approximation

Current bootstrapped ontology:
http://www.optique-project.eu/resource/FACTPAGES0791772669052272244620745164 1869/integrated_ontology1414664920484

OWL 2 profile	#Axioms	#Class	#Object properties	#Data properties
OWL 2 DL	3712	105	143	1172

Approximated ontology summary:

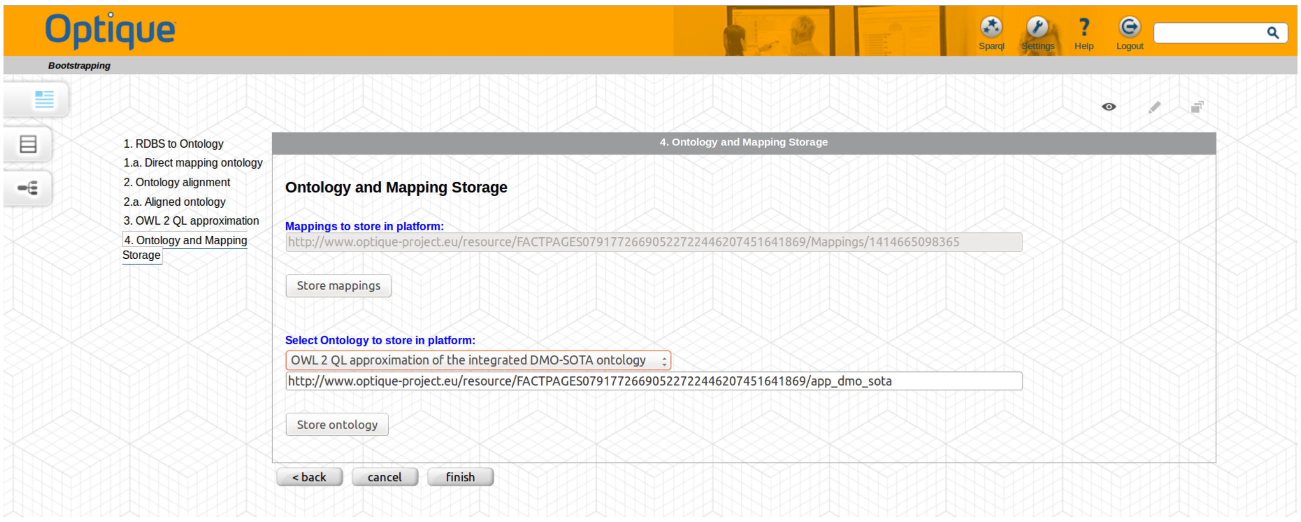
OWL 2 profile	#Total axioms	#Approximated axioms	#Discarded axioms
OWL 2 QL	3680	38	110

Approximation log:
Aproximation completed: Show Log Hide Log

Figure 7.4: O&M bootstrapper: OWL 2 QL approximation of the ontology

7.1.3 Ontology an Mapping Storage

As a last and optional step, the bootstrapped ontologies and mappings can be stored within the platform (see Figure 7.5).



1. RDBS to Ontology
1.a. Direct mapping ontology
2. Ontology alignment
2.a. Aligned ontology
3. OWL 2 QL approximation
4. Ontology and Mapping Storage

4. Ontology and Mapping Storage

Ontology and Mapping Storage

Mappings to store in platform:
http://www.optique-project.eu/resource/FACTPAGES07917726690522722446207451641869/Mappings/1414665098365

Store mappings

Select Ontology to store in platform:
OWL 2 QL approximation of the integrated DMO-SOTA ontology
http://www.optique-project.eu/resource/FACTPAGES07917726690522722446207451641869/app_dmo_sota

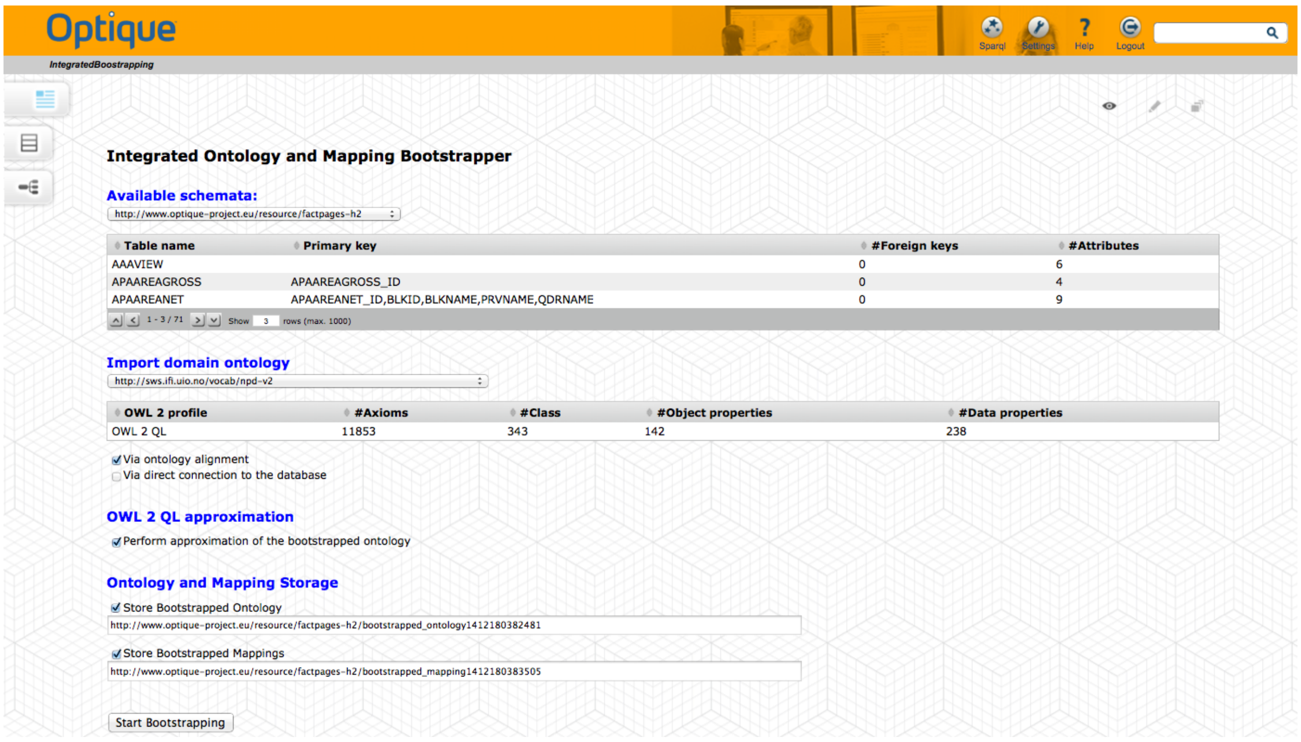
Store ontology

< back | cancel | finish

Figure 7.5: O&M bootstrapper: Ontology and mapping storage

7.1.4 Integrated O&M boostrapper

In addition to the step-by-step boostrapping, as described in previous sections, the O&M boostrapper also includes a one-step integrated bootstrapping interface (see Figure 7.6).



Integrated Ontology and Mapping Bootstrapper

Available schemata:
http://www.optique-project.eu/resource/factpages-h2

Table name	Primary key	#Foreign keys	#Attributes
AAAVIEW		0	6
APAAREAGROSS	APAAREAGROSS_ID	0	4
APAAREANET	APAAREANET_ID,BLKID,BLKNAME,PRVNAME,QDRNAME	0	9

1 - 3 / 71 Show 3 rows (max. 1000)

Import domain ontology
http://sws.ifi.uio.no/vocab/npd-v2

OWL 2 profile	#Axioms	#Class	#Object properties	#Data properties
OWL 2 QL	11853	343	142	238

✓ Via ontology alignment
☐ Via direct connection to the database

OWL 2 QL approximation

✓ Perform approximation of the bootstrapped ontology

Ontology and Mapping Storage

✓ Store Bootstrapped Ontology
http://www.optique-project.eu/resource/factpages-h2/bootstrapped_ontology1412180382481

✓ Store Bootstrapped Mappings
http://www.optique-project.eu/resource/factpages-h2/bootstrapped_mapping1412180383505

Start Bootstrapping

Figure 7.6: Widget for integrated bootstrapping.

Chapter 8

Evaluation

This section presents the evaluation conducted in the Statoil and Siemens scenarios.

8.1 Installing Optique Platform at Statoil

For reasons of confidentiality and consistency in presentation, these results are presented in Deliverable D9.2. A short summary follows:

8.1.1 The database

The experiments focused on parts of the *Exploration and Production Data Store* (EPDS), a corporate data store for subsurface data at Statoil. It has a complex schema, with ca. 1500 tables and as many views. The complexity of the schema, and the lack of documentation makes it impossible, other than for experts on the schema, to write correct SQL queries towards the database. Examples of why this is impossible is included in D9.2. We also have not seen evidence of anyone writing manual queries towards the database. They use an existing catalogue of predefined queries and tweak these as necessary. Advanced operations are done on the extracted data, and not by query combination or modification.

8.1.2 Experiments

We bootstrapped an ontology and mapping from the relevant parts of EPDS. The ontology contains 3274 classes, 3620 object properties, and 42308 datatype properties, The mappings comprise 3069 subjectMap-s and 43376 predicate ObjectMap-s, and all TripleMap-s select source data from exactly one database table.

In addition to the bootstrapped ontology, the Optique project has also produced two ontologies covering overlapping domains, the *Subsurface Exploration* ontology and the *NPD FactPages* ontology. The bootstrapped and developed ontologies have been aligned using the techniques described in Chapter 5. Furthermore, the Subsurface Exploration ontology has axioms outside OWL2 QL. The approximation techniques from Section 5.5 were applied to it. The results from alignment and approximation are included in D9.2.

The *coverage* of the terms in the query catalogue by the different ontologies was estimated using syntactic matching. For example, 15% terms in the query catalogue occur as classes in the bootstrapped ontology. The results are summarized in Figure 8.1. The experiments we conducted show that about half of the classes in the query catalogue are present in the alignment of the domain and bootstrapped ontology, and for most of these classes there are mappings to EPDS. Moreover, the majority of properties is also present in the alignment and most of them have mappings to EPDS.

8.2 Installing Optique Platform at Siemens

In this section we present our experience in running the O&M bootstrapping module in the Siemens use case. We start with a short description of database and the domain ontologies (see Deliverable 8.2 and [45] for more details), and we finally present a preliminary converage analysis of the bootstrapped ontology over the query terms.

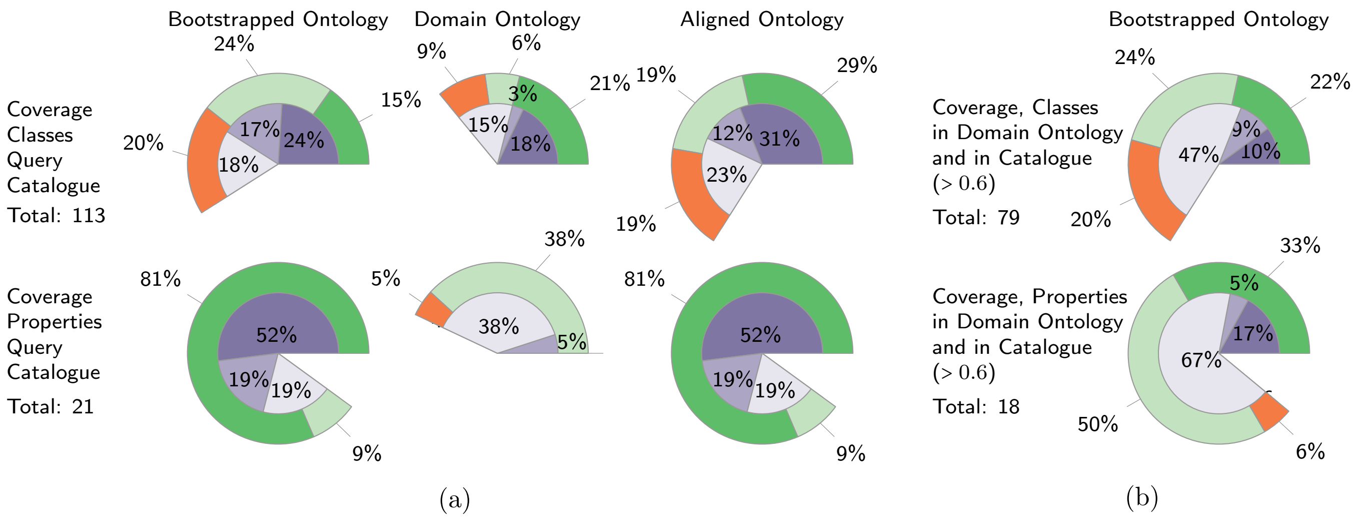


Figure 8.1: Inner pie charts show coverage by lexical confidence: ■ in [0.9, 1.0], ■ in [0.8, 0.9), □ in [0.6, 0.8). Outer pie charts represent the quality of the terms with a coverage above 0.6: 🟩 *true positive*, 🟩 *semi-true positive*, 🟧 *false positive*. Figure (a) displays the coverage of terms from the query catalogue with terms from ontologies; Figure (b) shows the overlap between terms from bootstrapped and imported ontologies that (with confidence > 0.6) occur in the query catalogue. Details in D9.2.

Ontology	Logical axioms	Classes	Object prop.	Datatype prop.
Bootstrapped	75	7	4	24
Diagnostic	107	31	11	7
Turbine	90	37	7	1

Table 8.1: Siemens ontology metrics

8.2.1 Siemens Schemata and Ontologies

The data in the Siemens use is stored in several databases with different schemata. Although the schemata are not specially large, the size of the data is in the order of hundreds of terabytes, e.g., there is about 15 GB of data associated to a single turbine, and it currently grows with the average rate of 30 GB per day [45].

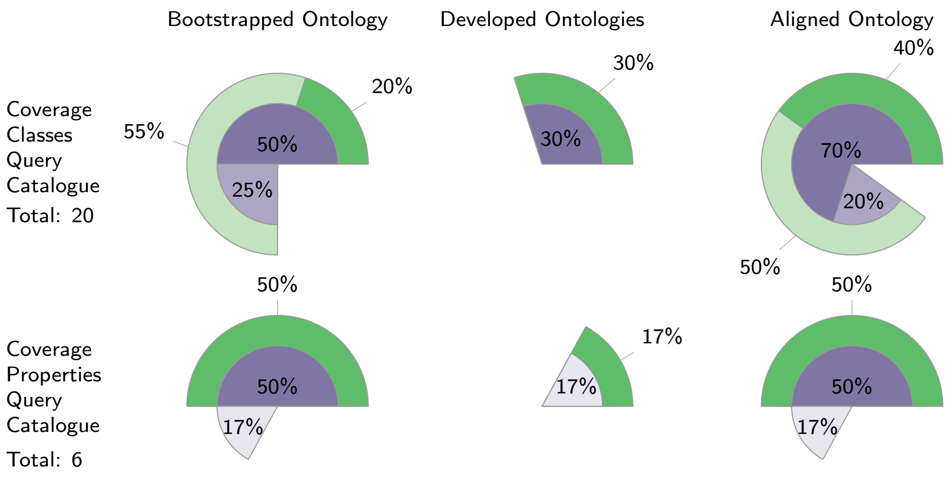
As for the Statoil use case, we boostrapped an ontology and mappings from one of the Siemens database schema (as described in Sections 5.2 and 5.3). In addition to the bootstrapped ontology, the Siemens ontology also contains two ontologies that have been developed for the Siemens use case: the *Diagnostic* and *Turbine* ontologies. The bootstrapped and developed ontologies have been align using the techniques described in Section 5.4. Details of the number of classes, properties, and axioms of these ontologies are in Table 8.1.

8.2.2 Coverage of Query Terms by the Ontologies

We extracted the terms from the current query catalogue which contains 27 query patterns. The terms has been split into query classes and query properties. In total, we identified 20 query classes and 6 query properties.

Figure 8.2 shows the coverage of the ontologies introduced above over the query terms: the upper three show the coverage of classes by, respectively (left-to-right) bootstrapped, developed, and aligned ontologies, the lower three show the coverage of properties.

Regarding the coverage of classes, the bootstrapped ontology covers 75% of the 20 classes occurring in the catalogue. Moreover, 50% of the query classes are matched to the bootstrapped ontology with a high lexical confidence, i.e., higher that 0.9. The coverage of query classes by the developed ontologies is lower than by the bootstrapped ontology, it is 30%. Regarding the coverage of properties, the bootstrapped ontology covers half of the properties occurring in the query catalogue, while there is a lower coverage by the developed ontologies. As in the Statoil use case, the bootstrapped ontology covers the query terms better that the developed ontologies. This can be interpreted as an indicatiog that the information needs with respect to the queries are semantically better reflected in the database schema than in the developed ontologies. Coverage of the aligned ontology for properties from the query catalogue is



(a)

Figure 8.2: Coverage of terms from the query catalogue with terms from ontologies. Inner pie charts show coverage by lexical confidence: ■ in [0.9, 1.0], ■ in [0.8, 0.9), □ in [0.6, 0.8). Outer pie charts represent the quality of the terms with a coverage above 0.6: 🟩 *true positive*, 🟩 *semi-true positive*.

the same as for the bootstrapped ontology, that is, all the properties covered by the domain ontology are also covered by the bootstrapped one. While the coverage of query classes reaches the 90%.

We also performed a manual assessment of every match for both query classes and properties. The results of out assessments are also in Figure 8.2 in the outer circles. For example, manual assessment of coverage of classes by the bootstrapped ontology gave 20% of true positives (i.e. correct matches) and 55% of semi-true positives (partially correct matches).

Chapter 9

Ongoing Work

9.1 Benchmark for Ontology Alignment

Figure 9.1 shows an OBDA scenario where the first ontology provides the vocabulary to formulate the queries (QF-Ontology) and the second is linked to the data and it is not visible to the users (DB-Ontology). Such OBDA scenarios is presented in real-world use cases such as in Optique (see Section 5.4). The integration via ontology alignment is required since only the vocabulary of the DB-Ontology is connected to the data.

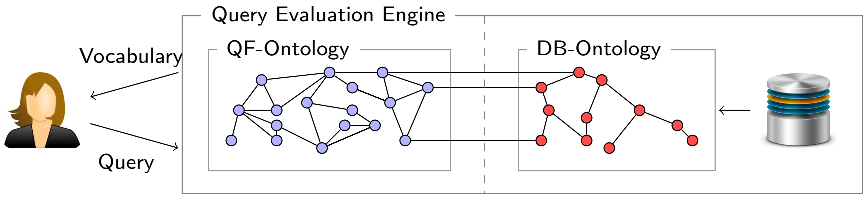


Figure 9.1: Ontology Alignment in an OBDA Scenario

The traditional benchmarks of the Ontology Alignment Evaluation Inititiative (OAEI) [5, 21] evaluate ontology matching systems w.r.t. scalability, multi-lingual support, instance matching, reuse of background knowledge, etc. Systems' effectiveness is, however, only assessed by means of classical information retrieval metrics (i.e, precision, recall and f-measure) w.r.t. a manually-curated reference alignment, provided by the organisers. However, query answering over aligned ontologies has not been addressed by any evaluation initiative so far.

We have introduced in the OAEI 2014 evaluation campaign the novel benchmark called Ontology Alignment for Query Answering[1] (OAQA) that aims at evaluating ontology matching systems w.r.t. the ability of the generated alignments to enable the answer of a set of queries in an OBDA scenario, where several ontologies exist [70] (see Appendix M for details).

9.2 Benchmark for Ontology and Mapping Bootstrapping

As presented in Section 2, there exist a number of approaches for the automatic bootstrapping of an ontology and RDB2RDF mappings from a relation database schema. However, no benchmarks to compare those approaches exist to date. In addition, slightly different assumptions about available input and expected output make existing systems difficult to compare on the same basis. This is also reflected by the fact that none of the aforementioned approaches did run comprehensive experiments directly comparing with any of the others.

An ongoing activity within Optique, is the creation of a comprehensive benchmark for measuring the quality of automatically generated RDB2RDF mappings. Our benchmark is based on realistic relational schemata and ontologies and assumes a whole set of different integration scenarios. To this end we also systematically analyze the characteristic available input information and requirements of different scenarios that call for the automatic or semiautomatic generation of RDB2RDF mappings.

[1]http://www.cs.ox.ac.uk/isg/projects/Optique/oaei/oa4qa/

9.3 Bootstrapping of Complex Mappings

In this chapter we will discuss a novel technique, which is currently under development, for bootstrapping complex, i.e., not direct, mappings. Now we describe two approaches to such a bootstrapping.

- Whenever it is suitable, we will consider the database schema as a directed graph with a node for each table, and an edge from table A to table B whenever A contains a foreign key referencing B. The basic idea of the approach is to translate every table into one or more classes. The table itself is considered as a superclass. To find its subclasses, one of the approaches would be to take joins of this table with the parent nodes. If these joins yield sufficiently different subsets of the tuples in the table, then they define subclasses. 'Sufficiently different' means a frequency cutoff of some kind. The reason for taking joins with parent nodes, rather than child nodes, is that a foreign key enforces a one-to-many relation.
- We can also look for sets of attributes in a table that have repeating values (see [41] for details). For example, if we have a table 'Person' with an attribute 'type', then particular (repeating) values of this attribute, e.g., 'student' or 'employee' can give rise to subclasses. A possible cutoff here is whether there are few values that occur: it may be undesirable to create too many subclasses.

Before going into more details, we provide some basic definitions.

9.3.1 Basic Definitions

First, we recall the basic notions about relational tables, which were introduced in Section 3.1), that we are going to use in this chapter. For ease of exposition, we assume that there is at most one foreign key between any two tables.

The following definitions declare a desired property of a bootstrapping complex mappings procedure.

Definition 9.3.1 *A bootstrapping procedure is an algorithm that, given a schema $\mathcal{S}$ and a database instance over $\mathcal{S}$, produces an ontology O and a set of mappings from $\mathcal{S}$ to O.*

Let $\mathcal{S}$ be a schema, O an ontology, M a set of mappings from $\mathcal{S}$ to O, and D a database instance over $\mathcal{S}$. We say that an *OBDA system* $(\mathcal{S}, M, O)$ is *consistent for* D, if $(D, \mathcal{S}, M, O) \not\models \bot$, where $\bot$ stands for falsehood. Note that all D, $\mathcal{S}$, M, and O are treated here as first-order formulae.

Definition 9.3.2 (Update-safety) *Let $\mathcal{S}$ be a schema, O an ontology, and M a set of mappings from $\mathcal{S}$ to O. We say that M and O are update-safe with respect to $\mathcal{S}$ if the OBDA system $(\mathcal{S}, M, O)$ is consistent for every database instance D over $\mathcal{S}$.*

A bootstrapping procedure is update-safe if it yields an update-safe set of mappings and ontology for every database instance of every schema.

In other words, an update-safe bootstrapping procedure does not add any constraints to the ontology that are not already enforced by the database schema. In particular, this means that the ontology to bootstrap will not "break" under updates to the data in the database. Note that this property is similar to the properties of information preservation and monotonicity identified by Sequeda et al. in [66] for direct mappings, however, it is weaker than either of them. Observe that in general, however, this property is rather restrictive: as databases frequently do not declare all the constraints that they satisfy, bootstrapping beyond this property can be desirable.

9.3.2 Finding classes based on joins

Let T be a table, and $\{S_1, \ldots, S_n\}$ be a set of tables that reference T. The basic idea is to create a subclass of T's base class for each S_i with $(T \ltimes_{\mathsf{fk}} S_i) \subset T$. See Algorithm 3 that implements this idea.

This basic idea, however, is complicated by the following. In relational databases, many-to-many relations between entities are usually represented by a separate table that contains only foreign keys, sometimes with an auto-incrementing integer primary key. Therefore, we prefer to map such tables to object properties, rather than classes. Thus, we need to develop means to effectively track and disregard such cases, which we leave for future work.

Algorithm 3: LJSubclasses

```
INPUT    : a table $T$, numbers $\alpha$ and $\beta$
OUTPUT: tables ResultTables
1 ResultTables = $\varnothing$
2 for each $T' \in \mathsf{refs}(T)$ do
3     $S \leftarrow T \ltimes_{\mathsf{fk}} T'$
4     if $\alpha < |S| < \beta$ then
5         ResultTables.add($S$)
6     end
7 end
8 return ResultTables
```

9.3.3 Finding classes based on clusters of attributes

Let T be a table, A the set of attributes in T, and V_a the set of different values for each attribute $a \in A$. We want to determine which rows of T form subclasses based on attribute similarity. To this end, we visualize the rows as points in space in the following natural manner. Each attribute $a \in A$ comprises one dimension and we associate to each value $v \in V_a$ a distinct integer. Thus, a row r in T is mapped to a tuple on the $|A|$-dimensional vector space whose components correspond to the values of r.

Example 9.3.1 *Given the following table and assuming we number the attribute values in the order in which they occur, rows 1 and 3 would be mapped to* $(1,1,1,1,1)$ *and* $(3,3,1,2,3)$*, respectively.*

id	*name*	*type*	*gender*	*phonenumber*
1	*alice*	*student*	*female*	*1234*
2	*bob*	*postdoc*	*male*	*2345*
3	*chris*	*student*	*male*	*3456*
4	*dora*	*prof*	*female*	*4567*

In order to determine which tuples are similar, we need to introduce a metric that measures distance in some manner. The standard Euclidean metric would not be prudent, since it makes no intuitive sense for the tuples $(1,2,3)$ and $(2,3,4)$ to have different distances to the origin. Intuitively, we want to count the number of values in which two points differ. This results in a sort of Hamming distance, which we define as

$$d(x,y) = \sum_{a \in A} \chi(x_a, y_a),$$

where $\chi(a,b) = 1$ if $a \neq b$ and $\chi(a,a) = 0$. This is fairly simple metric, which does not take into account the number of values that occur in an attribute column, i.e., it "punishes" differences in all columns equally (e.g., 'name' is given equal priority as 'gender' in Example 9.3.1). A way to circumvent this would be to define a variant of the above metric as follows:

$$d_w(x,y) = \sum_{a \in A} \frac{1}{|V_a|} \chi(x_a, y_a).$$

Of course, the weighting could be adjusted, depending on how much one wants to "punish" different values.

Example 9.3.2 *Given the table from Example 9.3.1 and the metrics defined above, rows 1 and 3 have the following distances:*

$$d(r_1,r_3) = 4$$
$$d_w(r_1,r_3) = \tfrac{1}{4} + \tfrac{1}{4} + 0 + \tfrac{1}{2} + \tfrac{1}{4} = \tfrac{5}{4}.$$

On the other hand, rows 1 and 4 have distances

$$d(r_1,r_4) = 4$$
$$d_w(r_1,r_4) = \tfrac{1}{4} + \tfrac{1}{4} + \tfrac{1}{3} + 0 + \tfrac{1}{4} = \tfrac{13}{12}.$$

We still need to see whether the metric d_w is worth using/investigating, but it should not be an issue to benchmark this once we have a working implementation.

In order to determine subclasses of a table, we can visualize the rows as point in space as discussed above. Using a suitable clustering algorithm, one can determine which points "belong together", i.e., which rows should belong to the same subclass.

Concerning the clustering algorithm, there are various options.

Hierarchical clustering

- divisive: Begin by considering all points as one cluster. Recursively divide each cluster into smaller clusters until we reach a point where we want to end or each point is its own cluster. This results in a tree-like hierarchy.

  Drawback: complex, doesn't solve the problem directly since we still need a "flat" clustering algorithm in each step, i.e., we need to decide how to split before we can continue with recursion.

- agglomerative: Begin with each point as its own cluster. Merge two clusters if they are each others nearest neighbor. Repeat this until there is only one cluster.

  Drawback: fairly memory intensive, but do not need a flat clustering algorithm. It is very likely that this approach will not be suitable for us, since we have to begin with too many clusters.

K-mediods clustering

- chooses k points which represent clusters and iteratively adds points to the clusters to which they are nearest.

  Drawback: we cannot use some standard techniques such as K-means since we do not use Euclidean distance.

As the next step, we will decide what approach to clustering will work the best in our framework.

Bibliography

[1] `http://jena.sourceforge.net/SquirrelRDF`.

[2] `http://www.w3.org/TR/r2rml/`.

[3] `http://www.w3.org/TR/rdb-direct-mapping/`.

[4] Rakesh Agrawal, Alexander Borgida, and H. V. Jagadish. Efficient Management of Transitive Relationships in Large Data and Knowledge Bases. In *ACM SIGMOD Conf. on Manag. of Data*, pages 253–262, 1989.

[5] J.L. Aguirre, Kai Eckert, Jérôme Euzenat, Alfio Ferrara, Willem Robert van Hage, Laura Hollink, Christian Meilicke, Andriy Nikolov, Dominique Ritze, François Scharffe, Pavel Shvaiko, Ondrej Sváb-Zamazal, Cássia Trojahn, E. Jiménez-Ruiz, Bernardo Cuenca Grau, and Benjamin Zapilko. Results of the Ontology Alignment Evaluation Initiative 2012. In *Ontology Matching Workshop*, 2012.

[6] Irina Astrova. Rules for mapping sql relational databases to owl ontologies. In *MTSR*, pages 415–424, 2007.

[7] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[8] Khalid Belhajjame, James Cheney, David Corsar, Daniel Garijo, Stian Soiland-Reyes, Stephan Zednik, Jun Zhao, Timothy Lebo, Satya Sahoo, and Deborah McGuinness. PROV-o: The PROV ontology. Technical report, World Wide Web Consortium, 2012.

[9] Alexandre Bertails and Eric Prud'hommeaux. Interpreting Relational Databases in the RDF Domain. In *K-CAP*, pages 129–136, 2011.

[10] C. Bizer and A. Seaborne. D2RQ-Treating non-RDF Databases as Virtual RDF Graphs. In *ISWC*, 2004.

[11] Alexander Borgida, Vinay K. Chaudhri, Paolo Giorgini, and Eric S. K. Yu, editors. *Conceptual Modeling: Foundations and Applications - Essays in Honor of John Mylopoulos*, volume 5600 of *Lecture Notes in Computer Science*. Springer, 2009.

[12] Alexander Borgida and Luciano Serafini. Distributed Description Logics: Assimilating Information from Peer Sources. *J. Data Sem.*, 1:153–184, 2003.

[13] Elena Botoeva, Diego Calvanese, and Mariano Rodriguez-Muro. Expressive approximations in DL-Lite ontologies. *Proceedings of the 14th International Conference on Artificial Intelligence: Methodology, Systems, Applications (AIMSA 2010)*, pages 21–31, 2010.

[14] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Antonella Poggi, Mariano Rodriguez-Muro, Riccardo Rosati, Marco Ruzzi, and Domenico Fabio Savo. The MASTRO system for ontology-based data access. *Semantic Web Journal*, 2(1):43–53, 2011.

[15] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tractable reasoning and efficient query answering in description logics: The dl-lite family. *Journal of Automated Reasoning*, 39(3):385–429, 2007.

[16] F. Cerbah and N. Lammari. *Perspectives in Ontology Learning*, chapter Ontology Learning from Databases: Some Efficient Methods to Discover Semantic Patterns in Data, pages 1–30. AKA / IOS Press. Serie, 2012.

[17] Farid Cerbah. Mining the content of relational databases to learn ontologies with deeper taxonomies. In *Web Intelligence*, pages 553–557, 2008.

[18] Cristina Civili, Marco Console, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Lorenzo Lepore, Riccardo Mancini, Antonella Poggi, Riccardo Rosati, Marco Ruzzi, Valerio Santarelli, and Domenico Fabio Savo. Mastro studio: Managing ontology-based data access applications. *PVLDB*, 6(12):1314–1317, 2013.

[19] Marco Console, Valerio Santarelli, and Domenico Fabio Savo. Efficient approximation in DL-Lite of OWL 2 ontologies. In *Proc. of the 26th Int. Workshop on Description Logic (DL)*, volume 1014 of *CEUR Electronic Workshop Proceedings,* `http://ceur-ws.org/`, pages 132–143, 2013.

[20] B. Cuenca Grau, I. Horrocks, B. Motik, B. Parsia, P. F. Patel-Schneider, and U. Sattler. OWL 2: The next step for OWL. *J. Web Sem.*, 6(4):309–322, 2008.

[21] Bernardo Cuenca Grau, Zlatan Dragisic, Kai Eckert, Jérôme Euzenat, Alfio Ferrara, Roger Granada, Valentina Ivanova, Ernesto Jiménez-Ruiz, Andreas Oskar Kempf, Patrick Lambrix, Andriy Nikolov, Heiko Paulheim, Dominique Ritze, François Scharffe, Pavel Shvaiko, Cássia Trojahn dos Santos, and Ondrej Zamazal. Results of the ontology alignment evaluation initiative 2013. In *Proceedings of the 8th International Workshop on Ontology Matching co-located with the 12th International Semantic Web Conference (ISWC 2013), Sydney, Australia, October 21, 2013.*, pages 61–100, 2013.

[22] Bernardo Cuenca Grau, Ian Horrocks, Yevgeny Kazakov, and Ulrike Sattler. Modular reuse of ontologies: Theory and practice. *J. Artif. Intell. Res.*, 31:273–318, 2008.

[23] Carlo Curino, Giorgio Orsi, Emanuele Panigati, and Letizia Tanca. Accessing and documenting relational databases through owl ontologies. In *FQAS*, pages 431–442, 2009.

[24] Jérôme David, Jérôme Euzenat, François Scharffe, and Cássia Trojahn. The Alignment API 4.0. *J. Sem. Web*, 2(1):3–10, 2011.

[25] Cristian Pérez de Laborda and Stefan Conrad. Database to Semantic Web Mapping Using RDF Query Languages. In *ER*, pages 241–254, 2006.

[26] María del Carmen Suárez-Figueroa, Asunción Gómez-Pérez, and Mariano Fernández-López. The neon methodology for ontology engineering. In María del Carmen Suárez-Figueroa, Asunción Gómez-Pérez, Enrico Motta, and Aldo Gangemi, editors, *Ontology Engineering in a Networked World*, pages 9–34. Springer, 2012.

[27] AnHai Doan, Alon Y. Halevy, and Zachary G. Ives. *Principles of Data Integration.* Morgan Kaufmann, 2012.

[28] William F. Dowling and Jean H. Gallier. Linear-Time Algorithms for Testing the Satisfiability of Propositional Horn Formulae. *J. Log. Prog.*, 1(3):267–284, 1984.

[29] Jérôme Euzenat. Semantic Precision and Recall for Ontology Alignment Evaluation. In *Int'l Joint Conf. on Artif. Intell. (IJCAI)*, pages 348–353, 2007.

[30] Daniel Faria, Ernesto Jimenez-Ruiz, Catia Pesquita, Emanuel Santos, and Francisco M. Couto. Towards annotating potential incoherences in BioPortal mappings. In *International Semantic Web Conference*, 2014.

[31] Matthew Fisher, Mike Dean, and Greg Joiner. Use of owl and swrl for semantic relational database translation. In *OWLED*, 2008.

[32] PREMIS Working Group et al. *Data dictionary for preservation metadata: final report of the PREMIS Working Group.* OCLC, 2005.

[33] Nicola Guarino. The ontological level: Revisiting 30 years of knowledge representation. In Borgida et al. [11], pages 52–67.

[34] Nicola Guarino and Christopher A. Welty. An overview of ontoclean. In Steffen Staab and Rudi Studer, editors, *Handbook on Ontologies*, International Handbooks on Information Systems, pages 151–172. Springer, 2004.

[35] Peter Haase et al. Optique System: Towards Ontology and Mapping Management in OBDA Solutions. In *Workshop on Debugging Ontologies and Ontology Mappings (WoDOOM)*, 2013.

[36] Alon Y. Halevy, Anand Rajaraman, and Joann J. Ordille. Data integration: The teenage years. In Umeshwar Dayal, Kyu-Young Whang, David B. Lomet, Gustavo Alonso, Guy M. Lohman, Martin L. Kersten, Sang Kyun Cha, and Young-Kuk Kim, editors, *VLDB*, pages 9–16. ACM, 2006.

[37] Ian Horrocks, Peter F. Patel-Schneider, and Frank van Harmelen. From $\mathcal{SHIQ}$ and RDF to OWL: the making of a web ontology language. *Journal of Web Semantics*, 1(1):7–26, 2003.

[38] Ernesto Jiménez-Ruiz and Bernardo Cuenca Grau. LogMap: Logic-based and Scalable Ontology Matching. In *Int'l Sem. Web Conf. (ISWC)*, pages 273–288, 2011.

[39] Ernesto Jiménez-Ruiz, Bernardo Cuenca Grau, Ian Horrocks, and Rafael Berlanga. Logic-based Assessment of the Compatibility of UMLS Ontology Sources. *J. Biomed. Semant.*, 2(Suppl 1):S2, 2011.

[40] Ernesto Jiménez-Ruiz, Bernardo Cuenca Grau, Yujiao Zhou, and Ian Horrocks. Large-scale Interactive Ontology Matching: Algorithms and Implementation. In *Eur. Conf. on Artif. Intell. (ECAI)*, pages 444–449, 2012.

[41] Sokratis Karkalas and Nigel J. Martin. Automatic semantic object discovery and mapping from non-normalised relational database systems. In *Advances in Information Systems, First International Conference, (ADVIS)* , pages 92–107, 2000.

[42] Yevgeny Kazakov, Markus Krotzsch, and Frantisek Simancik. Concurrent Classification of EL Ontologies. In *Int'l Sem. Web Conf. (ISWC)*, pages 305–320, 2011.

[43] E. Kharlamov, M. Giese, E. Jiménez-Ruiz, M. G. Skjæveland, A. Soylu, D. Zheleznyakov, T. Bagosi, M. Console, P. Haase, I. Horrocks, S. Marciuska, C. Pinkel, M. Rodriguez-Muro, M. Ruzzi, V. Santarelli, D. F. Savo, K. Sengupta, M. Schmidt, E. Thorstensen, J. Trame, and A. Waaler. Optique 1.0: Semantic Access to Big Data: The Case of Norwegian Petroleum Directorate's FactPages. In *International Semantic Web Conference (ISWC). Demo track*, 2013.

[44] Evgeny Kharlamov, Ernesto Jiménez-Ruiz, Dmitriy Zheleznyakov, et al. Optique: Towards OBDA Systems for Industry. In *Eur. Sem. Web Conf. (ESWC) Satellite Events*, pages 125–140, 2013.

[45] Evgeny Kharlamov, Nina Solomakhina, Ozgur Ozçep, Dmitriy Zheleznyakov, Thomas Hubauer, Steffen Lamparter, Mikhail Roshchin, and Ahmet Soylu. How semantic technologies can enhance data access at siemens energy. In *Proc. International Semantic Web Conference*, 2014.

[46] Phokion G. Kolaitis, Maurizio Lenzerini, and Nicole Schweikardt, editors. *Data Exchange, Integration, and Streams*, volume 5 of *Dagstuhl Follow-Ups*. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2013.

[47] Maurizio Lenzerini. Data integration: A theoretical perspective. In Lucian Popa, Serge Abiteboul, and Phokion G. Kolaitis, editors, *PODS*, pages 233–246. ACM, 2002.

[48] Dmitry V. Levshin. Mapping relational databases to the semantic web with original meaning. *Int. J. Software and Informatics*, 4(1):23–37, 2010.

[49] Man Li, Xiao-Yong Du, and Shan Wang. Learning ontology from relational database. In *Proceedings of International Conference on Machine Learning and Cybernetics*, 2005.

[50] Lina Lubyte and Sergio Tessaris. Automatic extraction of ontologies wrapping relational data sources. In *DEXA*, pages 128–142, 2009.

[51] Carsten Lutz, Inanç Seylan, and Frank Wolter. An automata-theoretic approach to uniform interpolation and approximation in the description logic $\mathcal{EL}$. In *Proc. of the 13th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR)*. AAAI Press, 2012.

[52] C. Meilicke. *Alignments Incoherency in Ontology Matching*. PhD thesis, University of Mannheim, 2011.

[53] Sergey Melnik, Hector Garcia-Molina, and Erhard Rahm. Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *ICDE*, pages 117–128, 2002.

[54] Boris Motik, Bernardo Cuenca Grau, Ian Horrocks, Zhe Wu, Achille Fokoue, and Carsten Lutz. OWL 2 Web Ontology Language – Profiles (2nd edition). W3C Recommendation, World Wide Web Consortium, December 2012. Available at `http://www.w3.org/TR/owl2-profiles/`.

[55] Boris Motik, Rob Shearer, and Ian Horrocks. Hypertableau Reasoning for Description Logics. *J. Artif. Intell. Res. (JAIR)*, 36:165–228, 2009.

[56] Jeff Z. Pan and Edward Thomas. Approximating OWL-DL ontologies. In *Proc. of the 22nd AAAI Conf. on Artificial Intelligence (AAAI)*, pages 1434–1439, 2007.

[57] C. Pinkel, C. Binnig, P. Haase, C. Martin, K. Sengupta, and J. Trame. How to Best Find a Partner? An Evaluation of Editing Approaches to Construct R2RML Mappings. In *ESWC*, 2014.

[58] C. Pinkel, C. Binnig, E. Kharlamov, and P. Haase. IncMap: Pay as You Go Matching of Relational Schemata to OWL Ontologies. In *Ontology Matching workshop*, 2013.

[59] Antonella Poggi, Domenico Lembo, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. Linking data to ontologies. *J. Data Semantics*, 10:133–173, 2008.

[60] Freddy Priyatna, Óscar Corcho, and Juan Sequeda. Formalisation and experiences of r2rml-based sparql to sql query translation using morph. In *WWW*, pages 479–490, 2014.

[61] Raymond Reiter. A Theory of Diagnosis from First Principles. *Artif. Intell.*, 32(1):57–95, 1987.

[62] Mariano Rodriguez-Muro, Roman Kontchakov, and Michael Zakharyaschev. Ontop at work. In *OWLED*, 2013.

[63] Heru Agus Santoso, Su-Cheng Haw, and Ziyad Abdul-Mehdi. Ontology extraction from relational database: Concept hierarchy as background knowledge. *Knowl.-Based Syst.*, 24(3):457–464, 2011.

[64] Stefan Schlobach. Debugging and Semantic Clarification by Pinpointing. In *Eur. Sem. Web Conf. (ESWC)*, pages 226–240. Springer, 2005.

[65] Stefan Schlobach and Ronald Cornet. Non-standard Reasoning Services for the Debugging of Description Logic Terminologies. In *Int'l Joint Conf. on Artif. Intell. (IJCAI)*, pages 355–362, 2003.

[66] Juan Sequeda, Marcelo Arenas, and Daniel P. Miranker. On directly mapping relational databases to rdf and owl. In *WWW*, pages 649–658, 2012.

[67] Juan Sequeda, Syed Hamid Tirmizi, Óscar Corcho, and Daniel P. Miranker. Survey of Directly Mapping SQL Databases to the Semantic Web. *Knowledge Eng. Review*, 26(4):445–486, 2011.

[68] Martin G. Skjæveland, Espen H. Lian, and Ian Horrocks. Publishing the Norwegian Petroleum Directorate's FactPages as Semantic Web Data. In *ISWC*, pages 162–177, 2013.

[69] A. Solimando, Ernesto Jiménez-Ruiz, and G. Guerrini. Detecting and Correcting Conservativity Principle Violations in Ontology-to-Ontology Mappings. In *International Semantic Web Conference*, 2014.

[70] Alessandro Solimando, Ernesto Jimenez-Ruiz, and Christoph Pinkel. Evaluating Ontology Alignment Systems in Query Answering Tasks. In *Poster paper at Int'l Sem. Web Conf. (ISWC)*, 2014.

[71] Ahmet Soylu, Martin Giese, Ernesto Jimenez-Ruiz, Guillermo Vega-Gorgojo, and Ian Horrocks. Experiencing optiquevqs: A multi-paradigm and ontology-based visual query system for end users. In *Under Review*, 2014.

[72] Dimitrios-Emmanuel Spanos, Periklis Stavrou, and Nikolas Mitrou. Bringing Relational Databases into the Semantic Web: A Survey. *Semantic Web*, 3(2):169–209, 2012.

[73] Mari Carmen Suárez-Figueroa, Asunción Gómez-Pérez, and Boris Villazón-Terrazas. How to write and use the ontology requirements specification document. In *On the Move to Meaningful Internet Systems: OTM 2009*, number 5871 in Lecture Notes in Computer Science, pages 966–982. Springer Berlin Heidelberg, January 2009.

[74] Tuvshintur Tserendorj, Sebastian Rudolph, Markus Krötzsch, and Pascal Hitzler. Approximate OWL-reasoning with Screech. In *Proc. of RR 2008*, pages 165–180. Springer, 2008.

[75] Holger Wache, Perry Groot, and Heiner Stuckenschmidt. Scalable instance retrieval for the semantic web by approximation. In *Proc. of WISE 2005*, pages 245–254. Springer, 2005.

Glossary

CQA	Consistent Query Answering
GSA	Global Semantic Approximation
IWB	Information Workbench
LSA	Local semantic approximation
NCS	Norwegian Continental Shelf
NPD	Norwegian Petroleum Directorate
O&M	Ontology and Mapping
OAEI	Ontology Alignment Evaluation Initiative
OBDA	Ontology-based Data Access
OWL	Web Ontology Language
QF	Query Formulation
QFI	Query Formulation Interface
R2RML	RDB to RDF Mapping Language
RDB	Relational Data Base
RDF	Resource Description Framework
REST	Representational State Transfer
URI	Uniform Resource Identifier
VM	Virtual Machine
W3C	World Wide Web Consortium
WP	Work Package

Appendix A

R2RML direct mapping cases

Table A.1 summarizes the cases that we have considered in the mapping bootstrapper. Note that some types of mappings have not been considered since they require (directly or indirectly) knowledge about the data (i.e. use of complex logical tables) or they have been left for the extended boostrapper (i.e. null treatment). Some other types of mappings are considered as optional like the generation of blank nodes when the primary key is missing.

Table A.1: R2RML direct mapping cases (http://www.w3.org/TR/rdb2rdf-test-cases/)

Type	**Identifier**	**Example DB**	**Mapping**	**Considered?**
Missing primary key	R2RMLTC0000	One table, a column, zero rows, no primary key	Direct mapping of an empty table	Yes
	R2RMLTC0001a	One table, one column, one row, no primary key	One column mapping, subject URI generation by using rr:template	Yes (default)
	R2RMLTC0001b	One table, one column, one row, no primary key	One column mapping, generation of a BlankNode subject by using rr:termType	Yes (opt.)
	R2RMLTC0002a	One table, two columns, one row, no primary key	Two columns mapping, generation of a subject URI by the concatenation of two column values	Yes (default)
	R2RMLTC0002b	One table, two columns, one row, no primary key	Two columns mapping, generation of a BlankNode subject by using rr:template and rr:termType	Yes (opt.)
	R2RMLTC0002d	One table, two columns, one row, no primary key	Two columns mapping, generation of a BlankNode subject by using a SQL Query that concatenates two columns	No
	R2RMLTC0003a	One table, three columns, one row, no primary key	Three columns mapping, undefined SQL Version identifier	No
	R2RMLTC0003b	One table, three columns, one row, no primary key	Three columns mapping, concatenation of columns, by using a rr:sqlQuery to produce literal	No

. . . continued

Type	**Identifier**	**Example DB**	**Mapping**	**Considered?**
	R2RMLTC0003c	One table, three columns, one row, no primary key	Three columns mapping, by using a rr:template to produce literal	No
	R2RMLTC0005	One table, three columns, three rows, two duplicate tuples, no primary key	Generation of BlankNodes from duplicate tuples	No
Generation of subject	R2RMLTC0006a	One table, one primary key, one column, one row	Long form of R2RML by using rr:constant in rr:subjectMap, rr:predicateMap, rr:objectMap and rr:graphMap	No
	R2RMLTC0007c	One table, one primary key, two columns, one row	One column mapping, using rr:class	Yes (default)
	R2RMLTC0007d	One table, one primary key, two columns, one row	One column mapping, specifying an rr:predicateObjectMap with rdf:type	Yes (opt.)
	R2RMLTC0007e	One table, one primary key, two columns, one row	One column mapping, using rr:graphMap and rr:class	No
	R2RMLTC0007f	One table, one primary key, two columns, one row	One column mapping, using rr:graphMap and specifying an rr:predicateObjectMap with rdf:type	No
Composite primary key	R2RMLTC0008a	One table, a composite primary key, three columns, one row	Generation of direct graph from a table with composite primary key	Yes (default)
	R2RMLTC0008b	One table, a composite primary key, three columns, one row	Generation of triples referencing object map	Yes (opt.?)
	R2RMLTC0008c	One table, a composite primary key, three columns, one row	Generation of triples by using multiple predicateMaps within a rr:predicateObjectMap	Yes (Opt.)
Foreign key management	R2RMLTC0009a	Two tables, a primary key, a foreign key	Generation of triples from foreign key relations	Yes (default)
	R2RMLTC0009b	Two tables, a primary key, a foreign key	Generation of triples to multiple graphs	No
	R2RMLTC0009c	Two tables, a primary key, a foreign key	Unnamed column in a logical table	No
	R2RMLTC0009d	Two tables, a primary key, a foreign key	Named column in logical table	No

...continued

Type	Identifier	Example DB	Mapping	Considered?
	DGraphTC0021	Two tables, two primary keys, a foreign key, references all nulls	Generation of triples for two tables, two primary keys, a foreign key, references all nulls	No
	DGraphTC0022	Two tables, a primary key, a foreign key, references no primary keys	Generation of triples from two tables, a primary key, a foreign key, references no primary keys	Yes
	DGraphTC0023	Two tables, two primary keys, two foreign keys, references to a key other than primary key	Generation of triples for two tables, two primary keys, two foreign keys, references to a key other than primary key	Yes
	DGraphTC0024	Two tables, two primary keys, a foreign key to a row with some NULLs in the key	Generation of triples from two tables, two primary keys, a foreign key to a row with some NULLs in the key	No
	DGraphTC0025	Three tables, three primary keys, three foreign keys	Generation of triples from three tables, three primary keys, three foreign keys	Yes
Many-to-Many tables	R2RMLTC0011b	Database with many to many relations	M to M relation, by using an additional Triples Map	Yes (default)
	R2RMLTC0011a	Database with many to many relations	M to M relation, by using a SQL query	No
IRI value in columns	R2RMLTC0014d	3 tables, one primary key, one foreign key	Test the translation of database type codes to IRIs	Yes (optional)
	R2RMLTC0019a	One table, one primary key, three columns, three rows	Generation of triples by using IRI value in columns	No
	R2RMLTC0019b	One table, one primary key, three columns, three rows	Generation of triples by using IRI value in columns, with data error	No
	R2RMLTC0020a	One table, one column, five rows	Generation of triples by using IRI value in columns	No
	R2RMLTC0020b	One table, one column, five rows	Generation of triples by using IRI value in columns, with data errors	No
Datatype management	R2RMLTC0016a	One table, one primary key, ten columns, three rows with sql datatypes	Table with datatypes: string and integer	Yes
	R2RMLTC0016b	One table, one primary key, ten columns, three rows with sql datatypes	Table with datatypes: real and float	Yes

... continued

Type	Identifier	Example DB	Mapping	Considered?
	R2RMLTC0016c	One table, one primary key, ten columns, three rows with sql datatypes	Table with datatypes: date and timestamp	Yes
	R2RMLTC0016d	One table, one primary key, ten columns, three rows with sql datatypes	Table with datatypes, boolean conversions	Yes
	R2RMLTC0016e	One table, one primary key, ten columns, three rows with sql datatypes	Table with datatypes, binary column	No
	R2RMLTC0018a	One table, one primary key, two columns, three rows	Generation of triples by using CHAR datatype column	Yes
Language management	R2RMLTC0015a	One table, three columns, one composite primary key, three rows, two languages	Generation of language tags from a table with language information	No
	R2RMLTC0015b	One table, three columns, one composite primary key, three rows, two languages	Generation of language tags from a table with language information, and a term map with invalid rr:language value	No
Special characters	R2RMLTC0010a	One table, a primary key, three columns, three rows	Template with table column with special chars	Yes
	R2RMLTC0010b	One table, a primary key, three columns, three rows	Template with table columns with special chars	Yes
	R2RMLTC0010c	One table, a primary key, three columns, three rows	Template with table columns with special chars and backslashes	No
	DGraphTC0017	I18N No Special Chars	I18N No Special Chars	No
rr:inverseExpression	R2RMLTC0014a	3 tables, one primary key, one foreign key	Subjectmap with rr:inverseExpression	No
	R2RMLTC0014b	3 tables, one primary key, one foreign key	Triplesmaps with rr:inverseExpression and rr:joinCondition	No
	R2RMLTC0014c	3 tables, one primary key, one foreign key	Triplesmaps with rr:inverseExpression, rr:joinCondition, and referencing object maps	No

Appendix B

OM 2013: IncMap

This appendix reports the paper:

– Christoph Pinkel, Carsten Binnig, Evgeny Kharlamov, Peter Haase IncMap: Pay as you go Matching of Relational Schemata to OWL Ontologies. In Proceedings of the International Ontology Matching workshop 2013

IncMap: Pay as you go Matching of Relational Schemata to OWL Ontologies

Christoph Pinkel[1], Carsten Binnig[2], Evgeny Kharlamov[3], and Peter Haase[1]

[1] fluid Operations AG, D-69190 Walldorf, Germany,
[2] University of Mannheim, D-68131 Mannheim, Germany,
[3] University of Oxford, Oxford, UK

Abstract. Ontology Based Data Access (OBDA) enables access to relational data with a complex structure through ontologies as conceptual domain models. A key component of an OBDA system are mappings between the schematic elements in the ontology and their correspondences in the relational schema. Today, in existing OBDA systems these mappings typically need to be compiled by hand, which is a complex and labor intensive task. In this paper we address the problem of creating such mappings and present *IncMap*, a system that supports a semi-automatic approach for matching relational schemata and ontologies. Our approach is based on a novel matching technique that represents the schematic elements of an ontology and a relational schema in a unified way. *IncMap* is designed to work in a query-driven, pay as you go fashion and leverages partial, user-verified mappings to improve subsequent mapping suggestions. This effectively reduces the overall effort compared to compiling a mappings in one step. Moreover, *IncMap* can incorporate knowledge from user queries to enhance suggestion quality.

1 Introduction

Effective understanding of complex data is a crucial task for enterprises to support decision making and retain competitiveness on the market. This task is not trivial especially since the data volume and complexity keep growing fast in the light of Big Data [1]. While there are many techniques and tools for scalable data analytics today, there is little known on how to find the *right* data.

Today, enterprise information systems of large companies store petabytes of data distributed across multiple – typically relational – databases, each with hundreds or sometimes even thousands of tables (e.g., [2]). For example, an installation of an SAP ERP system comes with tens of thousands of tables [3]. Due to the complexity of data a typical scenario for data analyses today involves a domain expert who formulates an analytical request and an IT expert who has to understand the request, find the data relevant to it, and then translate the request into an executable query. In large enterprises this process may iterate several times between the domain and IT experts, the complexity of data and other factors, and may take up to several weeks.

Ontology-based data access (OBDA) [4] is an approach that has recently emerged to provide semantic access to complex structured relational data. The core elements of an OBDA system are an *ontology*, describing the application domain, and a set of declarative *mappings*, relating the ontological schema elements (e.g., names of classes and properties) with the relational schema elements (e.g., names of table and attributes) of the underlying data sources. Using the ontology and the mappings, domain experts can access the data directly by formulating queries in terms defined in the ontology that reflects their vocabulary and conceptualization. Using query rewriting techniques, the end-user queries are then translated into queries over the underlying data sources.

Today, most approaches for ontology-based data access focus on the definition of mapping languages and the efficient translation of high-level user queries over an ontology into executable queries over relational data [4,5]. These approaches assume that a declarative mapping of the schema elements of the ontology to the relational elements is already given. So far, in real-world systems [6,7] that follow the ontology-based data access principle, the mappings have to be created manually. The costs for the manual creation of mappings constitute a significant entry barrier for applying OBDA in practice.

To overcome this limitation we propose a novel semi-automatic schema matching approach and a system called *IncMap* to support the creation of mappings directly from relational schemata to ontologies.

We focus on finding one-to-one (direct) correspondences of ontological and relational schema elements, while we also work on extensions for finding more complex correspondences. In order to compute mapping suggestions *IncMap* uses a relational schema, an OWL ontology, a set of user conjunctive queries over the ontology, and user feedback as basic input.

The matching approach of *IncMap* is inspired by the Similarity Flooding algorithm of Melnik et al. [8] that works well for schemata that follow the same modeling principles (e.g., same level of granularity). However, applying the Similarity Flooding algorithm naively for matching schema elements of a relational schema to an OWL ontology results in rather poor quality of the suggested correspondences as we show in our experiments. A major reason is the impedance mismatch between ontologies and relational schemata: While ontologies typically model high-level semantic information, relational schemata describe the syntactical structure on a very low level of granularity.

The contributions of the paper are the following:

– In Section 3, we propose a novel graph structure called *IncGraph* to represent schema elements from both ontologies and relational schemata in a unified way. Therefore, we devise algorithms to convert an ontology as well as a relational schema into their unified *IncGraph* representation. We also briefly discuss techniques to further improve *IncGraph*.
– In Section 4, we present our matching algorithm that we use for matching *IncGraph*s. Its most prominent feature is that *IncMap* can produce the mapping incrementally, query by query. While the original Similarity Flooding algorithm generates correspondences for all schema elements, *IncMap* supports a *pay as you go* matching strategy. For each query we produce only required mappings. *IncMap* leverages the structure of mappings from previ-

ous queries to improve suggestion quality. This effectively reduces the total effort for the user to verify mapping suggestions.
- Section 5 presents an experimental evaluation using different (real-world) relational schemata and ontologies. We see that even in the basic version of *IncMap*, the effort for creating a mapping is up to 20% less than using the Similarity Flooding algorithm in a naive way. In addition, the incremental version of *IncMap* can reduce the total effort by another 50% – 70%.

2 Background

In this section we briefly introduce ontologies [9], relational schemata, and the Similarity Flooding algorithm [8].

Ontologies. An ontology $\mathcal{O}$ specifies a conceptualization of a domain in terms of classes and properties and consists of a set of axioms. Without explanation, ontologies in this paper are OWL ontologies and we will use the following OWL constructs: object and data properties P, and domains $\texttt{Domain}(P)$ and ranges $\texttt{Range}(P)$ of properties. We denote with $\texttt{Class}(\mathcal{O})$ and $\texttt{Property}(\mathcal{O})$ the sets of class and property names, respectively, occurring in the ontology $\mathcal{O}$. For a given ontology $\mathcal{O}$, with $C \in \texttt{Domain}(P)$ we denote the fact that one can derive from $\mathcal{O}$ that the class name C is a domain of the property P. Also, $C' \in \texttt{Range}(P)$ denotes the fact that C' is a range of P and it is derivable from $\mathcal{O}$.

Relational Schemata. A relational schema $\mathcal{R}$ defines a set of relations (tables) T, where each table defines a set of columns c. We also assume that a schema contains foreign keys k that define references between tables.

Similarity Flooding Algorithm. The Similarity Flooding algorithm matches a given schema $\mathcal{S}$ with a schema $\mathcal{S}'$. In the first step, directed labeled graphs $\mathcal{G}(\mathcal{S})$ and $\mathcal{G}(\mathcal{S}')$ are constructed from $\mathcal{S}$ and $\mathcal{S}'$, where the nodes represent the schema elements, and the edges with labels define relationships between the schema elements. There is no exact procedure to construct the graphs from the schemata given in [8]. Thus, the Similarity Flooding algorithm is open for any graph construction process. The second step in the algorithm is to merge $\mathcal{G}(\mathcal{S})$ and $\mathcal{G}(\mathcal{S}')$ into one graph, a so-called pairwise connectivity graph $\texttt{PCG}$. Intuitively, each node of the $\texttt{PCG}$ is a pair of nodes, and represents a potential match between schema elements of $\mathcal{S}$ and $\mathcal{S}'$. Then, the $\texttt{PCG}$ is enriched with inverse edges and edge weights (propagation coefficients), where the value of the weights is based on the number of outgoing edges with the same label from a given node. This graph is called the induced propagation graph $\texttt{IPG}$. The final step of the algorithm is a fix-point computation to propagate initial similarities by using the structural dependencies represented by the propagation coefficients. The fix-point computation termination is based either on threshold values or the number of iterations. The result is a ranked list of suggested mappings. We refer to [8] for further details.

Algorithm 1: *IncGraph* for constructing graphs from ontologies

```
INPUT   : OWL ontology $\mathcal{O}$
OUTPUT: Graph $\mathcal{G} = (\mathcal{V}, \mathtt{Lbl}_\mathcal{V}, \mathcal{E}, \mathtt{Lbl}_\mathcal{E})$
1  Let $\mathcal{G} = (\mathcal{V}, \mathtt{Lbl}_\mathcal{V}, \mathcal{E}, \mathtt{Lbl}_\mathcal{E})$, $\mathcal{V} = \{n_\top\}$, $\mathtt{Lbl}_\mathcal{V} = \{(n_\top, \top)\}$, $\mathcal{E} = \emptyset$, $\mathtt{Lbl}_\mathcal{E} = \emptyset$;
2  foreach $C \in \mathtt{Class}(\mathcal{O})$ do $\mathcal{V} := \mathcal{V} \cup \{n_C\}$ and $\mathtt{Lbl}_\mathcal{E}(n_C) := C$
3  foreach $P \in \mathtt{Property}(\mathcal{O})$ do
4      $\mathcal{V} := \mathcal{V} \cup \{n_P\}$ and $\mathtt{Lbl}_\mathcal{V}(n_P) := P$; Let $C \in \mathtt{Domain}(P)$;
5      if $P$ is an object property then
6          $\mathcal{E} := \mathcal{E} \cup \{(n_C, n_P)\}$ and $\mathtt{Lbl}_\mathcal{V}((n_C, n_P)) :=$ 'ref';
7          Let $C' \in \mathtt{Range}(P)$;
8          $\mathcal{E} := \mathcal{E} \cup \{(n_P, n_{C'})\}$ and $\mathtt{Lbl}_\mathcal{V}((n_P, n_{C'})) :=$ 'ref';
9      else if $P$ is a data property then
10         $\mathcal{E} := \mathcal{E} \cup \{(n_C, n_P)\}$ and $\mathtt{Lbl}_\mathcal{E}((n_C, n_P)) :=$ 'value'
11 return $\mathcal{G}$.
```

Algorithm 2: *IncGraph* for constructing graphs from relational schemata

```
INPUT   : Relational Schema $\mathcal{R}$
OUTPUT: Graph $\mathcal{G} = (\mathcal{V}, \mathtt{Lbl}_\mathcal{V}, \mathcal{E}, \mathtt{Lbl}_\mathcal{E})$
1  Let $\mathcal{V} = \emptyset$, $\mathtt{Lbl}_\mathcal{V} = \emptyset$, $\mathcal{E} = \emptyset$, $\mathtt{Lbl}_\mathcal{E} = \emptyset$;
2  foreach table $T$ in $\mathcal{R}$ do
3      $\mathcal{V} := \mathcal{V} \cup \{n_T\}$ and $\mathtt{Lbl}_\mathcal{V}(n_T) := T$;
4      foreach column $c$ in $\mathcal{R}$ do
5          $\mathcal{V} := \mathcal{V} \cup \{n_c\}$ and $\mathtt{Lbl}_\mathcal{V}(n_c) := c$;
6          $\mathcal{E} := \mathcal{E} \cup \{(n_T, n_c)\}$ and $\mathtt{Lbl}_\mathcal{E}((n_T, n_c)) :=$ 'value'
7          if $c$ has a foreign key $k$ to some table $T'$ then
8              $\mathcal{V} := \mathcal{V} \cup \{n_k\}$ and $\mathtt{Lbl}_\mathcal{V}(n_k) := k$;
9              $\mathcal{E} := \mathcal{E} \cup \{(n_T, n_k)\}$ and $\mathtt{Lbl}_\mathcal{E}((n_T, n_k)) :=$ 'ref'
10             $\mathcal{E} := \mathcal{E} \cup \{(n_k, n_{T'})\}$ and $\mathtt{Lbl}_\mathcal{E}((n_k, n_{T'})) :=$ 'ref'
11 return $\mathcal{G}$.
```

3 The *IncGraph* Model

In this section, we describe the *IncGraph* model used by *IncMap* to represent schema elements of an OWL ontology $\mathcal{O}$ and a relational schema $\mathcal{R}$ in a unified way.

An *IncGraph* model is defined as directed labeled graph $\mathcal{G} = (\mathcal{V}, \mathtt{Lbl}_\mathcal{V}, \mathcal{E}, \mathtt{Lbl}_\mathcal{E})$. It can be used as input by the original Similarity Flooding algorithm (Section 2) or *IncMap*. $\mathcal{V}$ represents a set of vertices, $\mathcal{E}$ a set of directed edges, $\mathtt{Lbl}_\mathcal{V}$ a set of labels for vertices (i.e., one label for each vertex) and $\mathtt{Lbl}_\mathcal{E}$ a set of labels for edges (i.e., one label for each edge). A label $l_\mathcal{V} \in \mathtt{Lbl}_\mathcal{V}$ represents a name of a schema element whereas a label $l_\mathcal{E} \in \mathtt{Lbl}_\mathcal{E}$ is either "ref" representing a so called `ref`-edge or "value" representing a so called `val`-edge.

3.1 *IncGraph* Construction

The goal of the procedures for the basic construction is to incorporate explicit schema information from $\mathcal{O}$ and $\mathcal{R}$ into the *IncGraph* model. Incorporating implicit schema information is discussed in the next section.

Algorithm 1 creates an *IncGraph* model $\mathcal{G}$ for a given ontology $\mathcal{O}$. The algorithm constructs a vertex n_C for each class name $C \in \mathtt{Class}(\mathcal{O})$ and a vertex

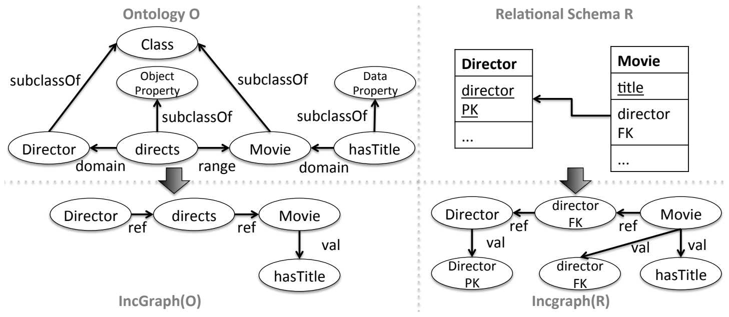


Fig. 1. *IncGraph* Construction Example

n_P for each property name $P \in \texttt{Property}(\mathcal{O})$ using the names of these ontology elements as label in $\texttt{Lbl}_\mathcal{V}$. Directed edges in the *IncGraph* model are created for each domain and range definition in $\mathcal{O}$. The labels $\texttt{Lbl}_\mathcal{E}$ for edges are either "ref" in case of an object property or "value" in case of a data property. For a domain definition in $\mathcal{O}$ the direction of the edge in $\mathcal{G}$ is from the node n_C representing the domain of P to the node n_P representing the property P. For a range definition the direction of the edge in $\mathcal{G}$ is from the node n_P representing object property to the node $n_{C'}$ representing the range of P (i.e., another class). If an object property in $\mathcal{O}$ has no range (respectively, domain) definition, then a directed labeled edge to a node $n_\top$ is added to explicitly model the most general range (respectively, domain), i.e., a top-level concept $\top$ like `Thing`.

Algorithm 2 creates a *IncGraph* model $\mathcal{G}$ for a given relational schema $\mathcal{R}$: The algorithm constructs a vertex n_T for each table and a vertex n_c for each column using the names of these schema elements as labels $\texttt{Lbl}_\mathcal{V}$. Directed edges with the label "value" are created from a node n_T representing a table to a node n_c representing a columns of that table. For columns with a foreign key k an additional node n_k is created. Moreover, two directed edges with the label "ref" are added, which represent a path from node n_T to a node $n_{T'}$ representing the referenced table via node n_k.

Figure 1 shows the result of applying these two algorithms to the ontology $\mathcal{O}$ and the relational schema $\mathcal{R}$ in this figure. Both $\mathcal{O}$ and $\mathcal{R}$ describe the same entities *Directors* and *Movies* using different schema elements. The resulting *IncGraph* models of $\mathcal{O}$ and $\mathcal{R}$ represent the schema structure in a unified way.

3.2 *IncGraph* Annotations

IncGraph is designed to represent both relational schemata and ontologies in a structurally similar fashion because matching approaches such as ours work best when the graph representations on both the source and target side are as similar as possible. However, even in *IncGraph* structural differences remain due to the impedance mismatch and different design patterns used in ontologies and relational schemata, respectively.

We consider this issue by supporting *annotations* in *IncGraph*. Annotations basically are additional `ref`-edges in either the source or target model that can be designed to bridge structural gaps for different design patterns or levels of granularity. For instance, *shortcut edges* in the relational *IncGraph* model could represent a multi-hop join over a chain of relationship relations. Annotations can be constructed by plug-ins during *IncGraph* construction.

We plan to evaluate the opportunities of different kinds of annotations in future work.

4 The *IncMap* System

In this section, we present our matching approach and system called *IncMap*. *IncMap* takes a source and target *IncGraph* as input, i.e., the *IncGraph*s produced for a relational schema and for an ontology as described in Section 3.

4.1 Overview of *IncMap*

In its basic version, *IncMap* applies the original Similarity Flooding algorithm (with minor adaptions) and thus creates initial mapping suggestions for the *IncGraph* of an ontology $\mathcal{O}$ and a relational schema $\mathcal{R}$. In its extended version, *IncMap* activates inactive `ref`-edges before executing the Similarity Flooding algorithm to achieve better mapping suggestions.

Another extension is the incremental version of *IncMap*. In this version the initial mapping suggestions are re-ranked by *IncMap* in a semi-automatic approach by including user feedback. Re-ranking works iteratively in a query-driven fashion thus increasing the quality of the suggested mappings. In each iteration, *IncMap* applies a version of the Similarity Flooding algorithm (as described before). However, in addition between each iteration user feedback is incorporated.

The idea of user feedback is that the user confirms those mapping suggestions of the previous iteration, which are required to answer a given user query over ontology $\mathcal{O}$. Confirmed suggestions are used as input for the next iteration to produce better suggestions for follow-up queries. This is in contrast to many other existing approaches (including the original Similarity Flooding algorithm) that return a mapping for the complete source and target schema only once.

IncMap is designed as a framework and provides different knobs to control which extensions to use and within each extension which concrete variants to choose (e.g., to select a concrete strategy for activating inactive edges). The goal of this section is to present *IncMap* with all its variants and to show their benefits for different real-world data sets in our experimental evaluation in Section 5. A major avenue of future work is to apply optimization algorithms to find the best configurations of *IncMap* for a given ontology $\mathcal{O}$ and schema $\mathcal{R}$ automatically by searching the configuration space based on the knobs presented before.

4.2 Basic Matching in *IncMap*

As already mentioned, in the basic version of *IncMap*, we simply apply the Similarity Flooding algorithm for the two *IncGraph*s produced for a relational schema $\mathcal{R}$ and for an ontology $\mathcal{O}$ similar to the process as described in Section 2.

As a first step, *IncMap* generates the `PCG` (i.e., a combined graph which pairs similar nodes of both input *IncGraph*s) using an initial lexical matching, which supports interchangeable matchers as one knob for configuration. One difference is the handling of inactive `ref`-edges in the input *IncGraphs*. For inactive `ref`-edges, which are not handled in the original Similarity Flooding, we apply the following rule when building the `PCG`: if an edge in the `PCG` refers to at least one inactive `ref`-edge in one of the *IncGraph* models, it also becomes inactive in the `PCG`.

In addition, other than in the original Similarity Flooding approach, where propagation coefficients for the `IPG` are ultimately determined during graph construction, our propagation coefficients can be calculated several times when the graph changes with the activation and deactivation of edges. Also, propagation coefficients in *IncMap* are modular and can be changed. In particular, a new weighting formula supported by *IncMap* considers the similarity scores on both ends of an edge in the `IPG`. The intuition behind this is that a higher score indicates better chances of the match being correct. Thus, an edge between two matches with relatively high scores is more relevant for the structure than an edge between one isolated well-scored match and another with a poor score. For calculating the weight $w(e)$ of a directed edge $e = (n_1, n_2)$ from n_1 to n_2 in the `IPG` where l is the label of the edge, we currently use two alternatives:

– *Original Weight as in [8]*: $w(e) = 1/out_l$ where out_l is the number of edges connected to node n_1 with the same label l
– *Normalized Similarity Product*: $w(e) = (score(n_1) * score(n_2))/out_l$.

4.3 Extended *IncMap*: Iterative User Feedback

Query-driven incremental mappings allow to leverage necessary user feedback after each iteration to improve the quality of mapping suggestions in subsequent iterations. One of the reasons why we have chosen Similarity Flooding as a basis for *IncMap* is the fact that user feedback can be integrated by adopting the initial match scores in an `IPG` before the fix-point computation starts.

Though the possibility of an incremental approach has been mentioned already in the Similarity Flooding paper [8], it so far has not been implemented and evaluated. Also, while it is simple to see *where* user feedback could be incorporated in the `IPG`, it is far less trivial to decide *which* feedback should be employed and *how* exactly it should be integrated in the graph. In this paper we focus on leveraging only the most important kind of user feedback, i.e., the previous confirmation and rejection of suggested mappings. We have devised three alternative methods how to add this kind of feedback into the graph.

First, as a confirmed match corresponds to a certain score of 1.0, while a rejected match corresponds to a score of 0.0, we could simply re-run the fix-point computation with adjusted initial scores of confirmed and/or rejected matches. We consequently name this first method *Initializer.* However, there is a clear risk that the influence of such a simple initialization on the resulting mapping is too small as scores tend to change rapidly during the first steps of the fix-point computation.

To tackle this potential problem, our second method guarantees maximum influence of feedback throughout the fix-point computation. Instead of just initializing a confirmed or rejected match with their final score once, we could repeat the initialization at the end of each step of the fix-point computation after normalization. This way, nodes with definite user feedback influence their neighborhood with their full score during each step of the computation. We therefore call this method *Self-Confidence Nodes*. However, as scores generally decrease in most parts of the graph during the fix-point computation and high scores become more important for the ranking of matches in later fix-point computation steps, this method implies the risk of over-influencing parts of the graph. For example, one confirmed match in a partially incorrect graph neighborhood would almost certainly move all of its neighbors to the top of their respective suggestion lists.

Finally, with our third method, we attempt to balance the effects of the previous two methods. We therefore do not change a confirmed match directly but include an additional node in `IPG` that can indirectly influence the match score during the fix-point computation. We name this method *Influence Nodes*. By keeping the scores of those additional influence nodes invariant we ensure permanent influence throughout all steps of the fix-point computation. Yet, the influence node only indirectly affects the neighborhood of confirmed nodes through the same propagation mechanism that generally distributes scores through the graph.

5 Experimental Evaluation

The main goal of *IncMap* is to reduce the human effort for constructing mappings between existing relational database schemata and ontologies. Mapping suggestions are intended to be used only after they have been validated by a user. Thus, there are two relevant evaluation measures: first, the percentage of the mappings in the reference mappings that *can be represented* by *IncMap*. We specify this percentage for all reference mappings when introducing them. Certain complex mappings (e.g., mappings performing data transformations) cannot be represented by *IncMap*. These complex mappings are rare in all real-world reference mappings we used in this paper. The second and most important measure is the *amount of work* that a user needs to invest to transform a set of mapping suggestions into the correct (intended) mappings. As the latter is the most crucial aspect, we evaluate our approach by measuring the work time required to transform our suggestions into the existing reference mappings.

5.1 Relational Schemata and Ontologies

To show the general viability of our approach, we evaluate *IncMap* in two scenarios with fairly different schematic properties. In addition to showing the key benefits of the approach under different conditions, this also demonstrates how the impact of modular parameters varies for different scenarios.

IMDB and Movie Ontology. As a first scenario, we evaluate a mapping from the schema of well known movie database IMDB[4] to the Movie Ontology [10]. With 27 foreign keys connecting 21 tables in the relational schema and 27 explicitly modeled object properties of 21 classes in the ontology, this scenario is average

[4] http://www.imdb.com

in size and structural complexity. The reference mappings we use to derive correspondences for this scenario[5] has been made available by the -ontop- team [11]. A set of example queries is provided together with these reference mappings. We use these to construct annotations for user queries as well as to structure our incremental, query-by-query experiments. We extract a total of 73 potential correspondences from this mapping, 65 of which can be represented by *IncMap* as mapping suggestions. This corresponds to 89% of the mappings that could be represented in *IncMap*.
MusicBrainz and Music Ontology. The second scenario is a mapping from the MusicBrainz database[6] to the Music Ontology [12]. The relational schema contains 271 foreign keys connecting 149 tables, while the ontology contains 169 explicitly modeled object properties and 100 classes, making the scenario both larger and more densely connected than the previous one. Here we use R2RML reference mappings that have been developed in the project EUCLID.[7] As there were no example queries provided with the mapping in this case, we use example queries provided by the Music Ontology for user query annotations and to structure the incremental experiment runs.

For these reference mappings, two out of 48 correspondences cannot be represented as mapping suggestions by *IncMap* as they require data transformations. This corresponds to 95.8% of the mappings that could be represented in *IncMap*.

5.2 Work Time Cost Model

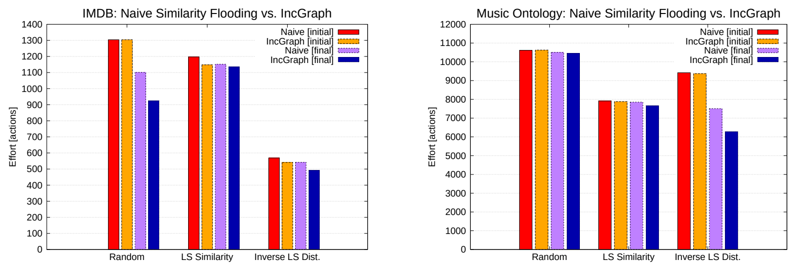
We evaluate our algorithms w.r.t. reducing *work time* (human effort). As the user feedback process always needs to transform mapping suggestions generated by *IncMap* into the correct mappings (i.e. to achieve a precision and recall of 100%), the involved effort is the one distinctive quality measure. To this end, we have devised a simple and straightforward work time cost model as follows: we assume that users validate mappings one by one, either accepting or rejecting them. We further assume that each validation, on average, takes a user the same amount of time $t_{validate}$. The costs for finding the correct correspondence for any concept in this case is identical with the rank of the correct mapping suggestion in the ranked list of mapping suggestions for the concept times $t_{validate}$.

As *IncMap* is interactive by design and would propose the user one mapping suggestion after another, this model closely corresponds to end user reality. We are aware that this process represents a simplification of mapping reality where users may compile some of the mappings by other means for various reasons. Nevertheless, this happens in the same way for any suggestion system and therefore does not impact the validity of our model for the purpose of comparison.

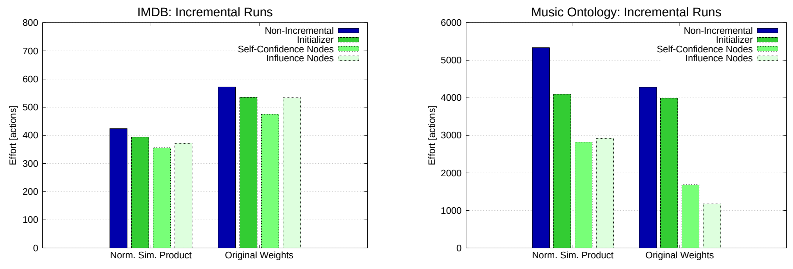
5.3 Experimental Evaluation

Experiment 1 – Naive vs. IncGraph*.* In our first experiment we compare the effort required to correct the mapping suggestions when the schema and ontol-

[5] https://babbage.inf.unibz.it/trac/obdapublic/wiki/Example_MovieOntology
[6] http://musicbrainz.org/doc/MusicBrainz_Database
[7] http://euclid-project.eu



(a) Naive vs. *IncGraph*



(b) Incremental Evaluation

Fig. 2. Experimental Evaluation

ogy are represented naively, or as *IncGraphs*. Additionally, we vary the lexical matcher used for the initial mapping between randomly assigned scores (minimal base line), Levenshtein similarity and inverse Levenshtein distance. Figure 2(a) shows that *IncGraph* in all cases works better than the naive approach. As *IncMap* reliably improves the mapping for all configurations, it also underlines the ability of *IncMap* to operate in a stable manner with different initial matchers.

Experiment 2 – Incremental Mapping Generation. Finally, we evaluated the best previous configurations incrementally, i.e., leveraging partial mappings. Figure 2(b) illustrates the effects on the total effort. We show total effort for all three incremental methods, for different propagation coefficients. Most significantly, incremental evaluation reduces the overall effort by up to 50% – 70%. More specifically, Self-Confidence Nodes and Influence nodes work much better than the naive Initializer approach.

6 Related Work

Many existing mapping systems rely on two-step mapping procedures: They employ lexical similarity of terms together with structural similarity of the structures ([13,14,15] or [16,17] for surveys). A very few of them rely on variations of Similarity Flooding to perform the latter task. However, to the best of our knowledge, all of these approaches focus on ontology-to-ontology rather than relational schema-to-ontology mappings. RiMOM [18] performs a multi-strategy mapping discovery between ontologies and performs mappings using a variant of the Similarity Flooding algorithm, while it relies on structural similarities of ontologies derived from sub-class and sub-property relationships, rather than connectivity of classes via properties as we do in order to get a better alignment of relational schemata and ontologies. In Yamm++ [19] the authors used Similarity Flooding and exploit both sub-class and sub-property relationships, and domain and ranges of ontologies, while they did it in a naive way which, as our experimental results showed, does not give good results for relational schemata-to-ontology mappings. Moreover, they use Similarity Flooding to obtain new mappings on top of the ones obtained via linguistic similarities, while we do not derive new mappings but refine the ranking over the linguistically derived ones. There are works on semi-automatic discovery of relational schema-to-ontology mappings, but they use approaches different from ours: For example, [20] transforms relational schemata and ontologies into directed labeled graphs respectively and reuse COMA [21] for essentially syntactic graph matching. Ronto [22] uses a combination of syntactic strategies to discover mappings by distinguishing the types of entities in relational schemata. The authors of [23] exploit structure of ontologies and relational schemata by calculating the confidence measures between virtual documents corresponding to them via the TF/IDF model. All these approaches do not incorporate implicit schema information and do not support an incremental mapping construction in the pay as you go fashion as *IncMap* does. Finally, [24] describes an approach to derive complex correspondences for a relational schema-to-ontology mapping using simple correspondences as input. This work is orthogonal to the approach presented in this paper.

7 Conclusions and Outlook

We presented *IncMap*, a novel semi-automatic matching approach for generating relational schema-to-ontology mappings. Our approach is based on a novel unified graph model called *IncGraph* for ontologies and relational schemata. *IncMap* implements a semi-automatic matching approach to derive mappings from *IncGraph*s using both lexical and structural similarities between ontologies and relational schemata. In order to find structural similarities *IncMap* exploits both explicit and implicit schema information. Moreover, *IncMap* allows to incorporate user queries and user feedback in an incremental way, thus, enabling a pay as you go fashion of the mapping generation. Our experiments with *IncMap* on different real-world relational schemata and ontologies showed that the effort for creating a mapping with *IncMap* is up to 20% less than using the Similarity Flooding algorithm in a naive way. The incremental version of *IncMap* reduces the total effort of mapping creation by another $50\% - 70\%$. As future work we plan to follow three lines: (1) add more implicit schema information (annotations) to the *IncGraph*s, (2) support more complex mappings in *IncMap*, and (3) devise a search strategy over the configuration space to auto-tune *IncMap*.

8 Acknowledgements

This work was supported by the Seventh Framework Program (FP7) of the European Commission under Grant Agreement 318338, the Optique project.

References

1. Beyer, M.A., Lapkin, A., Gall, N., Feinberg, D., Sribar, V.T.: 'Big Data' is Only the Beginning of Extreme Information Management. Gartner rep. G00211490 (2011)
2. Crompton, J.: Keynote talk at the W3C Workshop on Sem. Web in Oil & Gas Industry (2008) http://www.w3.org/2008/12/ogws-slides/Crompton.pdf.
3. SAP HANA Help: http://help.sap.com/hana/html/sql_export.html (2013)
4. Poggi, A., Lembo, D., Calvanese, D., Giacomo, G.D., Lenzerini, M., Rosati, R.: Linking Data to Ontologies. J. Data Semantics **10** (2008) 133–173
5. Kontchakov, R., Lutz, C., Toman, D., Wolter, F., Zakharyaschev, M.: The Combined Approach to Ontology-Based Data Access. In: IJCAI. (2011) 2656–2661
6. Hepp, M., Wechselberger, A.: OntoNaviERP: Ontology-Supported Navigation in ERP Software Documentation. In: International Semantic Web Conference. (2008)
7. Blunschi, L., Jossen, C., Kossmann, D., Mori, M., Stockinger, K.: SODA: Generating SQL for Business Users. PVLDB **5**(10) (2012) 932–943
8. Melnik, S., Garcia-Molina, H., Rahm, E.: Similarity Flooding: A Versatile Graph Matching Algorithm and its Application to Schema Matching. In: ICDE, IEEE Computer Society (2002)
9. Motik, B., Patel-Schneider, P.F., Parsia, B.: OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax (2012) W3C Rec.
10. Bouza, A.: MO – The Movie Ontology, http://www.movieontology.org (2010)
11. Rodriguez-Muro, M., Calvanese, D.: High Performance Query Answering over DL-Lite Ontologies. In: KR. (2012)
12. Raimond, Y., Giasson, F., (eds): Music Ontology, www.musicontology.com (2012)
13. Jiménez-Ruiz, E., Grau, B.C.: LogMap: Logic-Based and Scalable Ontology Matching. In: International Semantic Web Conference (1). (2011) 273–288
14. Lambrix, P., Tan, H.: SAMBO – A system for aligning and merging biomedical ontologies. J. Web Sem. **4**(3) (2006) 196–206
15. Fagin, R., Haas, L.M., Hernández, M.A., Miller, R.J., Popa, L., Velegrakis, Y.: Clio: Schema Mapping Creation and Data Exchange. In: Conceptual Modeling: Foundations and Applications. (2009) 198–236
16. Shvaiko, P., Euzenat, J.: Ontology Matching: State of the Art and Future Challenges. IEEE Trans. Knowl. Data Eng. **25**(1) (2013) 158–176
17. Rahm, E., Bernstein, P.A.: A Survey of Approaches to Automatic Schema Matching. In: VLDB J. (2001) 334–350
18. Li, J., Tang, J., Li, Y., Luo, Q.: RiMOM: A Dynamic Multistrategy Ontology Alignment Framework. IEEE Trans. Knowl. Data Eng. (2009) 1218–1232
19. Ngo, D., Bellahsene, Z.: YAM++: A Multi-strategy Based Approach for Ontology Matching Task. In: EKAW. (2012) 421–425
20. Dragut, E.C., Lawrence, R.: Composing Mappings Between Schemas Using a Reference Ontology. In: CoopIS/DOA/ODBASE (1). (2004) 783–800
21. Do, H.H., Rahm, E.: COMA – A System for Flexible Combination of Schema Matching Approaches. In: VLDB. (2002) 610–621
22. Papapanagiotou, P., Katsiouli, P., Tsetsos, V., Anagnostopoulos, C., Hadjiefthymiades, S.: Ronto: Relational to Ontology Schema Matching. In: AIS SIGSEMIS BULLETIN. (2006) 32–34
23. Hu, W., Qu, Y.: Discovering Simple Mappings Between Relational Database Schemas and Ontologies. In: ISWC/ASWC. (2007) 225–238
24. An, Y., Borgida, A., Mylopoulos, J.: Inferring Complex Semantic Mappings Between Relational Tables and Ontologies from Simple Correspondences. In: OTM Conferences (2). (2005)

Appendix C

Initial guidelines for OBDA specification

This appendix reports the paper:

– Maurizio Lenzerini, Riccardo Rosati, Valerio Santarelli, Domenico Fabio Savo. Towards a methodology for OBDA specification. Optique internal technical report, 2013.

Towards a methodology for OBDA specification

Maurizio Lenzerini, Riccardo Rosati. Valerio Santarelli, Domenico Fabio Savo

Dipartimento di Ingegneria Informatica, Automatica e Gestionale Antonio Ruberti
Sapienza Università di Roma
lenzerini,rosati,santarelli,savo@dis.uniroma1.it

Introduction

This documents aims at providing some initial guidelines towards the definition of a methodology for the specification of an ontology-based data access (OBDA) system.

Dealing with complex real world scenarios, such as those of businesses or enterprizes, even of small or medium scale, leads to facing several critical issues that are general enough to be taken into account in the OBDA approach regardless of the organization, and that can be summarized in:

(i) Knowledge gathering over the domain of interest;
(ii) Ontology development;
(iii) Data source analysis;
(iv) Mapping development.

Activities (i) and (iii) are requirement gathering activities, necessary to allow the ontology developer to acquire sufficient knowledge of the domain and of the data sources to develop, respectively, the ontology that models the domain of interest, and the mappings that link the ontology to the data sources. Both these activities require collaboration on the part of enterprise experts. Specifically, activity (i) requires the collaboration of domain experts, while activity (iii) requires the collaboration of IT experts, who have sufficient understanding of the data sources.

During the acquisition of the necessary information for the development of the ontology, it is crucial that the description of the domain be as independent as possible from the actual source database. The ontology is in fact supposed to model the domain of interest, and not the way the data sources that are used in the enterprise model such a domain.

Once the development of the ontology is considered to be sufficiently mature, the data source analysis phase can begin, which leads to the definition of the mappings. This activity can nevertheless lead to the acquisition of new knowledge of the domain, and to the consequent refinement of the ontology. The definition of the mappings, therefore, takes in input both the analysis of the data sources and the ontology. The result of this activity will be a set of mapping assertions.

The process which we have described is depicted in Figure 1.

In the following, we provide a detailed description of all the above-mentioned activities.

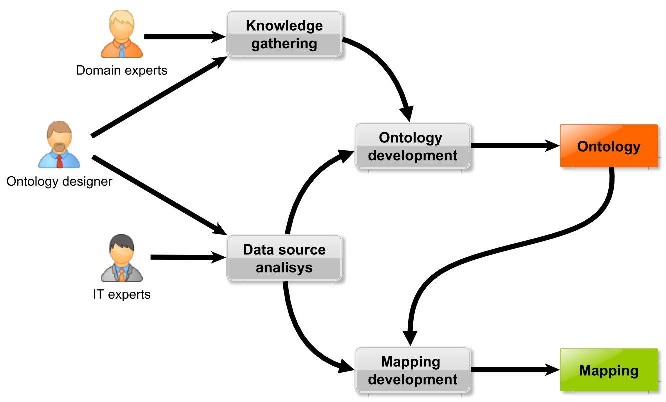


Fig. 1. The OBDA specification process.

Knowledge gathering over the domain of interest

When the OBDA approach is used in real-life projects, the first problem that needs to be addressed concerns communication with the domain expert. As for conceptual modeling in general, developing an ontology requires to exchange knowledge with people that are generally not expert of logical formalisms, but that have a deep understanding of the domain of interest. Exchanging this kind of knowledge requires the adoption of one or more techniques allowing the ontology developer to acquire the necessary knowledge for modeling the domain of interest in play.

Interviews. Interviews can be conducted by the ontology developer by following different approaches. The first, and most common, approach is the unstructured interview. In this scenario, the ontology developer and the knowledge expert freely discuss one or more aspects of the domain. This type of interview is typically conducted in the early stages of knowledge acquisition, when the ontology developer lacks sufficient knowledge to formulate a questionnaire. Instead, a semi-structured or structured interview consists of a series of pre-established questions, which the knowledge expert is provided with beforehand. In a semi-structured interview, the answers to these questions can lead to a discussion, during which the ontology developer focuses on some key aspects. In a structured interview, no follow-up discussion is expected. These two types of interviews are generally conducted in later stages of the project, when the ontology developer has a more mature understanding of the domain.

Commentary. This technique, instead of a discussion between the ontology developer and the domain expert, calls for the observation by the developer of the carrying out of a typical task by the knowledge expert. The expert is required to provide a running commentary of his thought process and of his activities as he completes a certain task. In order to avoid cognitive overload of the expert, who must both carry out his task and provide useful insight, it is common to have a second domain expert comment on the actions of another expert. This technique is quite useful because it allows the ontology designer to have a first-hand look at the actual behavior of the domain expert and of the tools at his disposal, instead of just listening to a recollection of his actions once a task is completed.

Teach Back. Once the ontology developer has acquired sufficient knowledge of a portion of the domain, he explains his understanding of that portion of the domain to the expert, which comments on it. This technique is very useful for to assess the level of comprehension of the developer, and to shed light on areas of confusion or misunderstanding.
For this activity, it is very important that the knowledge expert and the ontology developer find a communication tool with which both are comfortable. A technique that we have found to be very effective is the use of graphical formalisms to represent the ontology. In fact, a graphical representation is typically accessible to non-experts of logical and ontology formalisms, and can also allow the ontology developer to capture the main modeling features of the OWL 2 language. This choice proves to be effective, both in terms of improving communication between the parts, and also later phases of ontology refinement, allowing the developer to comfortably define and analyze the ontology.

Documentation Analysis. This technique calls for the inspection by the ontology expert of existing documentation of the business processes and the information systems that are used by the enterprise. This analysis allows the ontology developer to extract information regarding the domain of interest, both with respect to what data is used in these processes, and how it is managed by the information systems. Examples of aspects of which in-depth information can be acquired, through the inspection of such documentation, are: business entities, entity attributes, rules, and functionalities. This information allows the developer to carve a more precise outline of the domain that must be captured by the ontology.

Ontology development

There exist several approaches and methodologies for ontology engineering (e.g., [2], [4], [1], [5]). Based on the above results, in the following we briefly sketch some initial guidelines for this task. We remark, however, that we do not intend to propose any new approach in this direction: rather, our long-term goal is to create a methodology for OBDA specification that is independent on the particular methodology chosen for ontology development and maintenance.

The development of the ontology takes place during the requirement gathering phase. Initially, the acquired knowledge of the domain is inspected top-down, identifying the central concepts and the relations between them. Then, iteratively, with the collaboration of the domain expert, the ontology is refined. The knowledge gathered from the inspection of the data sources is used only in the final stages of the design phase, in order to refine the ontology.

As mentioned previously, in order to represent the ontology through a formalism which can both be comfortably understood by the domain expert and that is sufficiently expressive for the ontology designer, one can use a graphical language, which can be enriched with auxiliary documentation regarding the design choices that were made.

In case the ontology developer chooses to adopt this solution, the ontology must then be translated into a set of processable logical axioms. In other words, it must be written in some formal language, such as OWL 2. If an automated tool is available, it can be used to efficiently carry out this task. Otherwise, it must be done manually, with the aid of an ontology editor, such as Protègè.

The choice of the modeling language that will be adopted depends on the purpose for which the ontology is being developed. In fact, the use of very expressive languages such as the full OWL 2 language is very useful if the aim is to utilize the ontology as a formal description of the domain of interest. The expressivity of such languages allows the ontology designer to obtain a precise formalization of the domain. If instead the goal is to use the ontology for reasoning tasks, the high expressivity of the language used to model the ontology may be of hindrance. In particular, when wishing to access large quantities of data through the ontology, as in OBDA, the computational cost of very expressive languages such as OWL 2 is prohibitive. In these cases, it is necessary to recur to less expressive languages, thus resigning to having less complete representations of the domain of interest.

One possible solution to this issue is to allow the ontology designer the use of expressive languages to define ontologies that model the domain in great detail for the purpose of documentation and of other tasks that do not require strong computational effort, while adopting, for all those reasoning tasks in which particular computational properties are required, such as OBDA, descriptions of the domain of interest obtained through less expressive languages. The transition from an ontology formalized through an expressive language to one formalized through a less expressive language can be performed through automated approximation procedures.

During this phase, the execution of intentional reasoning tasks such as ontology classification is very important, in order to verify the quality and correctness of the choices made during ontology design.

Data source analysis

The first, crucial, step for this phase, is to identify the data sources which store the information regarding the domain that is represented by the developed ontology.

Once these data sources have been identified, it is necessary to understand where the information is stored and how it is possible to extract it. This task is very complex, due to the fact that often times the data sources are tailored towards being used by software applications, and that the gap between the database structure and the ontology representation of the domain can be quite large. For this reason, it is very important to obtain the collaboration of IT experts who have sufficient expertise of the data sources in this phase.

Other methods to acquire knowledge of the databases can be the inspection of documentation, and the analysis of the automated procedures that are used by the enterprizes to extract information.

Empirical evidence shows that, in order to gather sufficient information for defining the mappings, it is not sufficient to work solely with the database schema, but it is necessary to have access to the data.

Finally, it is quite common that this data source analysis phase can lead to acquiring new information that must be used to refine the ontology.

Mapping development

Mapping development has been studied in the fields of data integration and data exchange (see, e.g., [7], [3], [6], [8]). However, the issue of defining a comprehensive methodology for mapping development and mapping maintenance is still to be fully addressed.

When sufficient information of the data sources has been gathered, and one is confident that the development of the ontology is sufficiently stable, the mapping development phase can begin. Currently, this task is completely performed manually, and involves both the ontology designer and the IT expert.

To successfully carry out this task, it is important to have access to the data sources, and in particular, to be able to query the database, in order to be able to verify the mapping assertions that are being defined.

Our experience teaches us that in this phase, automated systems for the verification of the syntactic correctness of the mappings are very useful. Furthermore, reasoning services over the OBDA system, such as the consistency checking service, can highlight issues that can be indicators of errors that were made during mapping specification.

References

1. A. Borgida, V. K. Chaudhri, P. Giorgini, and E. S. K. Yu, editors. *Conceptual Modeling: Foundations and Applications - Essays in Honor of John Mylopoulos*, volume 5600 of *Lecture Notes in Computer Science*. Springer, 2009.
2. M. del Carmen Suárez-Figueroa, A. Gómez-Pérez, and M. Fernández-López. The neon methodology for ontology engineering. In M. del Carmen Suárez-Figueroa, A. Gómez-Pérez, E. Motta, and A. Gangemi, editors, *Ontology Engineering in a Networked World*, pages 9–34. Springer, 2012.
3. A. Doan, A. Y. Halevy, and Z. G. Ives. *Principles of Data Integration*. Morgan Kaufmann, 2012.
4. N. Guarino. The ontological level: Revisiting 30 years of knowledge representation. In Borgida et al. [1], pages 52–67.
5. N. Guarino and C. A. Welty. An overview of ontoclean. In S. Staab and R. Studer, editors, *Handbook on Ontologies*, International Handbooks on Information Systems, pages 151–172. Springer, 2004.
6. A. Y. Halevy, A. Rajaraman, and J. J. Ordille. Data integration: The teenage years. In U. Dayal, K.-Y. Whang, D. B. Lomet, G. Alonso, G. M. Lohman, M. L. Kersten, S. K. Cha, and Y.-K. Kim, editors, *VLDB*, pages 9–16. ACM, 2006.
7. P. G. Kolaitis, M. Lenzerini, and N. Schweikardt, editors. *Data Exchange, Integration, and Streams*, volume 5 of *Dagstuhl Follow-Ups*. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2013.
8. M. Lenzerini. Data integration: A theoretical perspective. In L. Popa, S. Abiteboul, and P. G. Kolaitis, editors, *PODS*, pages 233–246. ACM, 2002.

Appendix D

LogMap: OM 2013 paper

This appendix reports the paper:

– Ernesto Jiménez-Ruiz, Bernardo Cuenca Grau, and Ian Horrocks. LogMap and LogMapLt Results for OAEI 2013. OM 2013.

LogMap and LogMapLt results for OAEI 2013

Ernesto Jiménez-Ruiz, Bernardo Cuenca Grau, Ian Horrocks

Department of Computer Science, University of Oxford, Oxford, UK

Abstract. We present the results obtained in the OAEI 2013 campaign by our ontology matching system LogMap and its 'lightweight" variant called LogMapLt. The LogMap project started in January 2011 with the objective of developing a scalable and logic-based ontology matching system. This is our fourth participation in the OAEI and the experience has so far been very positive.

1 Presentation of the system

LogMap [11, 12] is a highly scalable ontology matching system with built-in reasoning and inconsistency repair capabilities. LogMap also supports (real-time) user interaction during the matching process, which is essential for use cases requiring very accurate mappings. LogMap is one of the few ontology matching system that (1) can efficiently match semantically rich ontologies containing tens (and even hundreds) of thousands of classes, (2) incorporates sophisticated reasoning and repair techniques to minimise the number of logical inconsistencies, and (3) provides support for user intervention during the matching process. LogMap is also available as a "lightweight" variant called LogMapLt, which essentially only applies (efficient) string matching techniques.

LogMap relies on the following elements, which are keys to its favourable scalability behaviour (see [11, 12] for details).

Lexical indexation. An inverted index is used to store the lexical information contained in the input ontologies. This index is the key to efficiently computing an initial set of mappings of manageable size. Similar indexes have been successfully used in information retrieval and search engine technologies [4].

Logic-based module extraction. The practical feasibility of unsatisfiability detection and repair critically depends on the size of the input ontologies. To reduce the size of the problem, we exploit ontology modularisation techniques. Ontology modules with well-understood semantic properties can be efficiently computed and are typically much smaller than the input ontology (e.g. [7]).

Propositional Horn reasoning. The relevant modules in the input ontologies together with (a subset of) the candidate mappings are encoded in LogMap using a Horn propositional representation. Furthermore, LogMap implements the classic Dowling-Gallier algorithm for propositional Horn satisfiability [8, 10]. Such encoding, although incomplete, allows LogMap to detect unsatisfiable classes soundly and efficiently.

Axiom tracking and greedy repair. LogMap extends Dowling-Gallier's algorithm to track all mappings that may be involved in the unsatisfiability of a class. This extension is key to implementing a highly scalable repair algorithm.

Semantic indexation. The Horn propositional representation of the ontology modules and the mappings are efficiently indexed using an interval labelling schema [1] — an optimised data structure for storing directed acyclic graphs (DAGs) that significantly reduces the cost of answering taxonomic queries [6, 17]. In particular, this semantic index allows us to answer many entailment queries over the input ontologies and the mappings computed thus far as an index lookup operation, and hence without the need for reasoning. The semantic index complements the use of the propositional encoding to detect and repair unsatisfiable classes.

1.1 Adaptations made for the 2013 evaluation

The new version of LogMap also integrates MORe [2, 3] as OWL 2 reasoner. MORe is a modular reasoner which combines a fully-fledged (and slower) reasoner with a profile specific (and more efficient) reasoner.

LogMap's algorithm described in [11–13] has also been adapted to meet the requirements of the new interactive matching track which uses an *Oracle* as expert user.

LogMap aims at making a reduced number of calls to the Oracle, i.e.: only those borderline mappings that cannot be clearly included or excluded with automatic heuristics. For each call to the Oracle, LogMap applies conflict and ambiguity based heuristics (see [12] for details) to reduce the remaining number of calls (i.e. mappings).

Additionally, the interactive algorithm described in [12] has been slightly extended to include object and data properties in the process.

1.2 Link to the system and parameters file

LogMap is open-source and released under GNU Lesser General Public License 3.0.[1] Latest components and source code are available from the LogMap's Google code page: `http://code.google.com/p/logmap-matcher/`.

LogMap distributions can be easily customized through a configuration file containing the matching parameters.

LogMap, including support for interactive ontology matching, can also be used directly through an AJAX-based Web interface: `http://csu6325.cs.ox.ac.uk/`. This interface has been very well received by the community, with more than 900 requests processed so far coming from a broad range of users.

1.3 Modular support for mapping repair

Only very few systems participating in the OAEI 2013 competition implement repair techniques. As a result, existing matching systems (even those that typically achieve very high precision scores) compute mappings that lead in many cases to a large number of unsatisfiable classes.

We believe that these systems could significantly improve their output if they were to implement repair techniques similar to those available in LogMap. Therefore, with

[1] `http://www.gnu.org/licenses/`

Table 1: Results for Benchmark track.

System	biblio 2012			biblioc		
	P	R	F	P	R	F
LogMap	1.00	0.47	0.64	0.73	0.42	0.53
LogMapLt	0.95	0.50	0.66	0.43	0.50	0.46

Table 2: Results for Anatomy track.

System	P	R	F	Time (s)
LogMap	0.918	0.846	0.881	13
LogMapLt	0.962	0.728	0.829	7

the goal of providing a useful service to the community, we have made LogMap's ontology repair module (LogMap-Repair) available as a self-contained software component that can be seamlessly integrated in most existing ontology matching systems [14].

2 Results

In this section, we present a summary of the results obtained by LogMap and LogMapLt in the OAEI 2013 campaign. Please refer to `http://oaei.ontologymatching.org/2013/results/index.html` for complete results.

2.1 Benchmark track

Ontologies in this track have been synthetically generated. The goal of this track is to evaluate the matching systems in scenarios where the input ontologies lack important information (e.g., classes contain no meaningful URIs or labels) [9].

Table 1 summarises the average results obtained by LogMap and LogMapLt. Note that the computation of candidate mappings in LogMap and LogMapLt heavily relies on the similarities between the vocabularies of the input ontologies; hence, there is a direct negative impact in the cases where the labels are replaced by random strings.

2.2 Anatomy track

This track involves the matching of the Adult Mouse Anatomy ontology (2,744 classes) and a fragment of the NCI ontology describing human anatomy (3,304 classes). The reference alignment has been manually curated [19], and it contains a significant number of non-trivial mappings.

Table 2 summarises the results obtained by LogMap and LogMapLt. LogMap ranked 3rd among the systems not using specialised background knowledge. Regarding mapping coherence, only two tools (including LogMap) generated coherent alignments. The evaluation was run on a server with 3.46 GHz (6 cores) and 8GB RAM.

Table 3: Results for Conference track.

System	**RA1 reference**			**RA2 reference**			**Time (s)**
	P	**R**	**F**	**P**	**R**	**F**	
LogMap	0.80	0.59	0.68	0.76	0.54	0.63	24
LogMapLt	0.73	0.50	0.59	0.68	0.45	0.54	21

Table 4: Results for Library track.

System	**P**	**R**	**F**	**Time (s)**
LogMap	0.777	0.645	0.705	99
LogMapLt	0.646	0.771	0.703	20

2.3 Conference track

The Conference track uses a collection of 16 ontologies from the domain of academic conferences [18]. These ontologies have been created manually by different people and are of very small size (between 14 and 140 entities). The track uses two reference alignments RA1 and RA2. RA1 contains manually curated mappings between 21 ontology pairs, while RA2 also contains composed mappings based on the alignments in RA1.

Table 3 summarises the average results obtained by LogMap and LogMapLt. The last column represents the total runtime on generating all 21 alignments. Tests were run on a laptop with Intel Core i5 2.67GHz and 8GB RAM. LogMap ranked 3rd and produced coherent alignments.

2.4 Multifarm track

This track is based on the translation of the OntoFarm collection of ontologies into 9 different languages [16]. Both LogMap and LogMapLt, as expected, obtained poor results since they do not implement specific multilingual techniques.

2.5 Library track

The library track involves the matching of the STW thesaurus (6,575 classes) and the TheSoz thesaurus (8,376 classes). Both of these thesauri provide vocabulary for economic and social sciences. Table 4 summarises the results obtained by LogMap and LogMapLt. The track was run on a computer with one 2.4GHz core with 7GB RAM and 2 cores. LogMap ranked 5th in this track.

2.6 Interactive matching track

The interactive track is based on the conference track and it uses the RA1 reference alignment as Oracle. Table 5 summarizes the obtained results by LogMap with and without the interactive mode activated. LogMap with interactivity (LogMap-Int) improved both the average Precision and Recall wrt LogMap with the interactive mode

Table 5: Results for Interactive track.

System	**RA1 reference**			**Calls**	**Time (s)**
	P	**R**	**F**		
LogMap	0.80	0.59	0.68	0	24
LogMap-Int	0.90	0.64	0.73	91	27

Table 6: Summary results for the Large BioMed track

System	**Total Time (s)**	**P**	**R**	**F**	**Inc. Degree.**
LogMap-BK	2,391	0.904	0.700	0.785	0.013%
LogMap	2,485	0.910	0.689	0.780	0.015%
LogMapLt	371	0.874	0.517	0.598	34.1%

deactivated, and it only performed 91 calls to the Oracle along the 21 matching tasks (i.e. less than 5 questions per ontology pair).

Not that, although LogMap-Int ranked 1st in the interactive matching track, it could not outperform the best tool in the conference track, which obtained a F-measure of 0.74 (wrt the RA1 reference alignment). Nevertheless, there is still room for improvement and we aim at implementing more sophisticated matching and interactive techniques.

2.7 Large BioMed track

This track consists of finding alignments between the Foundational Model of Anatomy (FMA), SNOMED CT, and the National Cancer Institute Thesaurus (NCI). These ontologies are semantically rich and contain tens of thousands of classes. UMLS Metathesaurus [5] has been selected as the basis for the track reference alignments.

In this track LogMap has been evaluated with two variants: LogMap and LogMap-BK. LogMap-BK uses normalisations and spelling variants from the general (biomedical) purpose UMLS Lexicon,[2] while LogMap has this feature deactivated.

Table 6 summarises the results obtained by LogMap and LogMapLt. The table shows the total time in seconds to complete all tasks in the track and averages for Precision, Recall, F-measure and Incoherence degree. The track was run on a server with 16 CPUs and allocating 15GB RAM.

Regarding mapping coherence, only two tools (including LogMap and its variant LogMap-BK) generated almost coherent alignments. LogMap-BK ranked 3rd among the systems not using specialised background knowledge and 1st among the systems computing almost coherent alignments. LogMapLt was the fastest to complete all tasks.

[2] `http://www.nlm.nih.gov/pubs/factsheets/umlslex.html`

Table 7: Results for Instance matching track.

System	RDFT		
	P	R	F
LogMap	0.922	0.746	0.812

2.8 Instance matching

This year only LogMap participated in the Instance Matching track. The dataset was based on dbpedia ontology[3] and included controlled transformations in the data (i.e. value and structure transformations).

Table 7 summarises the average results obtained by LogMap. The results are quite promising considering that LogMap does not implement sophisticated instance matching techniques. Furthermore, LogMap outperformed one of the participating tools specialised in instance matching.

Adaptations to the original dataset The original provided dataset was preprocessed in order to be properly interpreted by the OWL API and to avoid inconsistencies when reasoning. Next we summarise the performed changes:

– *Added import of dbpedia:* The dataset (ABOX) is based on dbpedia, however, the dbpedia ontology was not included as TBOX. Hence the OWL API was interpreting the instance entities of the dataset as "annotations" and not as "OWL named individuals". Furthermore, by adding dbpedia TBOX to the datasets, an OWL 2 reasoner could be used to infer the corresponding class type for each instance.
– *Minor changes to dbpedia:* The integration of the provided dataset (ABOX) and dbpedia (TBOX) resulted in an inconsistent knowledge base. The inconsistencies were due to some data property assertion axioms pointing to the incorrect datatype and a functional datatype property which was used in two or more data property assertion axioms with the same subject. To avoid these inconsistencies dbpedia was slightly modified by removing the range and the functionality of the corresponding data properties.
– *Added additional object properties:* The dataset also references the object properties "curriculum", "places" and "label" which are not included in the dbpedia ontology. Hence, these properties has been explicitly declared as OWL object properties.
– *Removal of invalid characters:* the dataset also included some characters that could not be processed by the OWL API and Protégé (e.g. \u).

3 General comments and conclusions

3.1 Comments on the results

LogMap, apart from Benchmark and Multifarm tracks for which does not implement specific techniques, has been one of the top systems in the OAEI 2013. Furthermore,

[3] `http://dbpedia.org/`

it has also been one of the few systems implementing repair techniques and providing (almost) coherent mappings in all tracks.

LogMap's main weakness relies on the fact that the computation of candidate mappings is based on the similarities between the vocabularies of the input ontologies; hence, there is a direct negative impact in the cases where the ontologies are lexically disparate or do not provide enough lexical information (e.g. Benchmark and Multifarm).

3.2 Discussions on the way to improve the proposed system

LogMap is now a stable and mature system that has been made available to the community. There are, however, many exciting possibilities for future work. For example we aim at exploiting background knowledge to be competitive in the Multifarm track and to improve the performance in the other tracks.

3.3 Comments on the OAEI test cases

The number and quality of the OAEI tracks is growing year by year. However, there is always room for improvement:

Comments on the OAEI instance matching track. I consider the 2012 IIMB Instance Matching track more challenging, from the logical point of view, than the current task. The IIMB dataset included a TBOX and the controlled transformations also involved changes on the instance class types. Thus the application of logic based techniques had an important impact since lexically similar instances belonging to two disjoint class types should not be matched.

Comments on the OAEI interactive matching track. The new interactive track has been a very important step forward in the OAEI, however, larger and more challengings tasks should be included. For example, matching tasks (e.g. anatomy and largebio) where the number of questions to the expert user or Oracle may be critical. Furthermore, it is quite unlikely that the expert user will be perfect, thus, the interactive matching track should also consider the evaluation of several Oracles with different error rates such as the evaluation performed in [12].

Comments on the OAEI largebio track. One of the objectives of the largebio track is the creation of a "silver standard" reference alignment by harmonising the output of the different participating systems. In the next OAEI campaign it would be very interesting to actively use this "silver standard" in the construction of the track's reference alignment.

3.4 Comments on the OAEI 2013 measures

Although the *mapping coherence* is a measure already used in the OAEI we consider that is not given yet the required weight in the evaluation. Thus, developers focus on creating matching systems that maximize the F-measure but they disregard the impact of the generated output in terms of logical errors. As a result, even highly precise mappings lead to a large number of unsatisfiable classes.

Thus, we encourage ontology matching system developers to develop their own repair techniques or to use state-of-the-art techniques such as Alcomo [15] and LogMap-Repair (see Section 1.3), which have shown to work well in practice [14].

Acknowledgements

This work was supported by the Seventh Framework Program (FP7) of the European Commission under Grant Agreement 318338, ”Optique”, the Royal Society, and the EPSRC projects Score!, ExODA and MaSI[3].

References

1. Agrawal, R., Borgida, A., Jagadish, H.V.: Efficient management of transitive relationships in large data and knowledge bases. In: ACM SIGMOD Conf. on Management of Data. pp. 253–262 (1989)
2. Armas Romero, A., Cuenca Grau, B., Horrocks, I.: MORe: Modular Combination of OWL Reasoners for Ontology Classification. In: Int’l Sem. Web Conf. (ISWC). pp. 1–16 (2012)
3. Armas Romero, A., Cuenca Grau, B., Horrocks, I., Jiménez-Ruiz, E.: MORe: a Modular OWL Reasoner for Ontology Classification. In: OWL Reasoning Evaluation (ORE) (2013)
4. Baeza-Yates, R.A., Ribeiro-Neto, B.A.: Modern Information Retrieval. ACM Press / Addison-Wesley (1999)
5. Bodenreider, O.: The unified medical language system (UMLS): integrating biomedical terminology. Nucleic Acids Research 32, 267–270 (2004)
6. Christophides, V., Plexousakis, D., Scholl, M., Tourtounis, S.: On labeling schemes for the Semantic Web. In: Int’l World Wide Web (WWW) Conf. pp. 544–555 (2003)
7. Cuenca Grau, B., Horrocks, I., Kazakov, Y., Sattler, U.: Modular reuse of ontologies: Theory and practice. J. Artif. Intell. Res. 31, 273–318 (2008)
8. Dowling, W.F., Gallier, J.H.: Linear-time algorithms for testing the satisfiability of propositional Horn formulae. J. Log. Prog. 1(3), 267–284 (1984)
9. Euzenat, J., Rosoiu, M.E., dos Santos, C.T.: Ontology matching benchmarks: Generation, stability, and discriminability. J. Web Sem. 21, 30–48 (2013)
10. Gallo, G., Urbani, G.: Algorithms for testing the satisfiability of propositional formulae. J. Log. Prog. 7(1), 45–61 (1989)
11. Jiménez-Ruiz, E., Cuenca Grau, B.: LogMap: Logic-based and Scalable Ontology Matching. In: Int’l Sem. Web Conf. (ISWC). pp. 273–288 (2011)
12. Jiménez-Ruiz, E., Cuenca Grau, B., Zhou, Y., Horrocks, I.: Large-scale interactive ontology matching: Algorithms and implementation. In: European Conf. on Artif. Intell. (ECAI). pp. 444–449 (2012)
13. Jiménez-Ruiz, E., Grau, B.C., Horrocks, I.: LogMap and LogMapLt results for OAEI 2012. In: Proceedings of the 7th International Workshop on Ontology Matching (2012)
14. Jimenez-Ruiz, E., Meilicke, C., Cuenca Grau, B., Horrocks, I.: Evaluating mapping repair systems with large biomedical ontologies. In: 26th Description Logics Workshop (2013)
15. Meilicke, C.: Alignment Incoherence in Ontology Matching. Ph.D. thesis, University of Mannheim (2011)
16. Meilicke, C., Castro, R.G., Freitas, F., van Hage, W.R., Montiel-Ponsoda, E., de Azevedo, R.R., Stuckenschmidt, H., Šváb-Zamazal, O., Svátek, V., Tamilin, A., Trojahn, C., Wang, S.: MultiFarm: a benchmark for multilingual ontology matching. J. Web Sem. (2012)
17. Nebot, V., Berlanga, R.: Efficient retrieval of ontology fragments using an interval labeling scheme. Inf. Sci. 179(24), 4151–4173 (2009)
18. Šváb, O., Svátek, V., Berka, P., Rak, D., Tomášek, P.: OntoFarm: towards an experimental collection of parallel ontologies. In: Int’l Sem. Web Conf. (ISWC). Poster Session (2005)
19. Zhang, S., Mork, P., Bodenreider, O.: Lessons learned from aligning two representations of anatomy. In: Conf. on Principles of Knowledge Representation and Reasoning (KR) (2004)

Appendix E

LogMap: OM 2014 paper

This appendix reports the paper:

– Ernesto Jiménez-Ruiz, Bernardo Cuenca Grau, Weiguo Xia, Alessandro Solimando, Xi Chen, Valerie Cross, Yuan Gong, Shuo Zhang, Anurekha Chennai-Thiagarajan. OM 2014.

LogMap family results for OAEI 2014 [⋆]

E. Jiménez-Ruiz[1], B. Cuenca Grau[1], W. Xia[2], A. Solimando[3], X. Chen[2], V. Cross[2], Y. Gong[1], S. Zhang[1], and A. Chennai-Thiagarajan[2]

[1] Department of Computer Science, University of Oxford, Oxford UK
[2] Computer Science and Software Engineering, Miami University, Oxford, OH, United States
[3] Dipartimento di Informatica, Università di Genova, Italy

Abstract. We present the results obtained in the OAEI 2014 campaign by our ontology matching system LogMap and its variants: LogMap-C, LogMap-Bio and LogMapLt. The LogMap project started in January 2011 with the objective of developing a scalable and logic-based ontology matching system. This is our fifth participation in the OAEI and the experience has so far been very positive.

1 Presentation of the system

Ontology matching systems typically rely on lexical and structural heuristics and the integration of the input ontologies and the mappings may lead to many undesired logical consequences. In [13] three principles were proposed to minimize the number of potentially unintended consequences, namely: *(i) consistency principle*, the mappings should not lead to unsatisfiable classes in the integrated ontology; *(ii) locality principle*, the mappings should link entities that have similar *neighbourhoods*; *(iii) conservativity principle*, the mappings should not introduce alterations in the classification of the input ontologies. Violations to these principles may hinder the usefulness of ontology mappings. The practical effect of these violations, however, is clearly evident when ontology alignments are involved in complex tasks such as query answering [17].

LogMap [12, 14] is a highly scalable ontology matching system that implements the consistency and locality principles. LogMap also supports (real-time) user interaction during the matching process, which is essential for use cases requiring very accurate mappings. LogMap is one of the few ontology matching system that *(i)* can efficiently match semantically rich ontologies containing tens (and even hundreds) of thousands of classes, *(ii)* incorporates sophisticated reasoning and repair techniques to minimise the number of logical inconsistencies, and *(iii)* provides support for user intervention during the matching process.

LogMap relies on the following elements, which are keys to its favourable scalability behaviour (see [12, 14] for details).

Lexical indexation. An inverted index is used to store the lexical information contained in the input ontologies. This index is the key to efficiently computing an initial set of mappings of manageable size. Similar indexes have been successfully used in information retrieval and search engine technologies [2].

[⋆] This research was financed by the Optique project with the grant agreement FP7-318338.

Logic-based module extraction. The practical feasibility of unsatisfiability detection and repair critically depends on the size of the input ontologies. To reduce the size of the problem, we exploit ontology modularisation techniques. Ontology modules with well-understood semantic properties can be efficiently computed and are typically much smaller than the input ontology (e.g. [6]).

Propositional Horn reasoning. The relevant modules in the input ontologies together with (a subset of) the candidate mappings are encoded in LogMap using a Horn propositional representation. Furthermore, LogMap implements the classic Dowling-Gallier algorithm for propositional Horn satisfiability [7]. Such encoding, although incomplete, allows LogMap to detect unsatisfiable classes soundly and efficiently.

Axiom tracking and greedy repair. LogMap extends Dowling-Gallier's algorithm to track all mappings that may be involved in the unsatisfiability of a class. This extension is key to implementing a highly scalable repair algorithm.

Semantic indexation. The Horn propositional representation of the ontology modules and the mappings are efficiently indexed using an interval labelling schema [1] — an optimised data structure for storing directed acyclic graphs (DAGs) that significantly reduces the cost of answering taxonomic queries [5, 19]. In particular, this semantic index allows us to answer many entailment queries over the input ontologies and the mappings computed thus far as an index lookup operation, and hence without the need for reasoning. The semantic index complements the use of the propositional encoding to detect and repair unsatisfiable classes.

1.1 Adaptations made for the 2014 evaluation

In the OAEI 2014 campaign we have participated with 3 additional variants:

LogMapLt is a "lightweight" variant of LogMap, which essentially only applies (efficient) string matching techniques.

LogMap-C is a variant of LogMap which, in addition to the consistency and locality principles, also implements the conservativity principle (see details in [21, 20]). The repair algorithm is more aggressive than in LogMap, thus we expect highly precise mappings but with a significant decrease in recall.

LogMap-Bio includes an extension to use BioPortal [10, 11] as a (dynamic) provider of mediating ontologies instead of relying on a few preselected ontologies [4]. In the OAEI 2014, LogMap-Bio uses the top-5 mediating ontologies given by the algorithm presented in [4]. Note that, LogMap-Bio only participates in the biomedical tracks. In the other tracks the results are expected to be the same as LogMap.

LogMap's algorithm described in [12, 14] has also been adapted with the following new functionalities:

i **Multilingual support.** We have implemented a multilingual module based on *google translate*[4] to participate in the Multifarm track. Additionally, in order to split Chi-

[4] Currently we use the (unofficial) API available at `https://code.google.com/p/google-api-translate-java/`.

nese words, we rely on the ICTCLAS library[5] developed by the Institute of Computing Technology of the Chinese Academy of Sciences.

ii **Extended repair algorithm.** We have extended the Horn propositional projection of the input ontologies to involve data and object properties in the repair process [24]. LogMap's repair module is now more complete and it is also able to repair (object and data) property mappings.[6]

iii **Extended interactive support.** The interactive algorithm described in [14] has been slightly extended to include object and data properties in the process. Note that this extension was already included in the OAEI 2013 campaign.

1.2 Link to the system and parameters file

LogMap is open-source and released under GNU Lesser General Public License 3.0.[7] Latest components and source code are available from the LogMap's Google code page: `http://code.google.com/p/logmap-matcher/`.

LogMap distributions can be easily customized through a configuration file containing the matching parameters.

LogMap, including support for interactive ontology matching, can also be used directly through an AJAX-based Web interface: `http://csu6325.cs.ox.ac.uk/`. This interface has been very well received by the community, with more than 1,500 requests processed so far coming from a broad range of users.

1.3 Modular support for mapping repair

Only very few systems participating in the OAEI competition implement repair techniques. As a result, existing matching systems (even those that typically achieve very high precision scores) compute mappings that lead in many cases to a large number of unsatisfiable classes.

We believe that these systems could significantly improve their output if they were to implement repair techniques similar to those available in LogMap. Therefore, with the goal of providing a useful service to the community, we have made LogMap's ontology repair module (LogMap-Repair) available as a self-contained software component that can be seamlessly integrated in most existing ontology matching systems [15, 9].

2 Results

In this section, we present a summary of the results obtained by the LogMap family in the OAEI 2014 campaign. Please refer to `http://oaei.ontologymatching.org/2014/results/index.html` for complete results.

[5] `https://code.google.com/p/ictclas4j/`

[6] The OAEI 2014 coherence results does not exhibit these improvements since only the conference track ontologies involve mappings among properties and LogMap 2013 was already coherent. It does have, however, an impact when repairing other mapping sets as shown in [24].

[7] `http://www.gnu.org/licenses/`

Table 1: Results for Benchmark track.

System	biblio			cose			dog		
	P	F	R	P	F	R	P	F	R
LogMap	0.40	0.40	0.40	0.38	0.41	0.45	0.96	0.15	0.08
LogMap-C	0.42	0.41	0.40	0.39	0.41	0.43	0.98	0.15	0.08
LogMapLt	0.43	0.46	0.50	0.37	0.43	0.50	0.86	0.71	0.61

Table 2: Results for Anatomy track.

System	P	F	R	Time (s)
LogMap-Bio	0.888	0.897	0.906	535
LogMap	0.918	0.881	0.846	12
LogMap-C	0.975	0.802	0.682	22
LogMapLt	0.962	0.829	0.728	5

2.1 Benchmark track

Ontologies in this track have been synthetically generated. The goal of this track is to evaluate the matching systems in scenarios where the input ontologies lack important information (e.g., classes contain no meaningful URIs or labels) [8].

Table 1 summarises the average results obtained by LogMap and its variants. Note that the computation of candidate mappings in LogMap (and its variants) heavily relies on the similarities between the vocabularies of the input ontologies; hence, there is a direct negative impact in the cases where the labels are replaced by random strings. Surprisingly, LogMapLt obtained the best results in the dog test case.

2.2 Anatomy track

This track involves the matching of the Adult Mouse Anatomy ontology (2,744 classes) and a fragment of the NCI ontology describing human anatomy (3,304 classes). The reference alignment has been manually curated [25], and it contains a significant number of non-trivial mappings.

Table 2 summarises the results obtained by the LogMap family. LogMap-Bio ranked 2nd in the track. The use of BioPortal as mediating ontology provider had a significant improvement in recall. LogMap-Bio runtime is near 10 minutes since the discovery of the mediating ontologies is performed on-the-fly [4]. Regarding mapping coherence, only two tools (apart from LogMap, LogMap-C and LogMap-Bio) generated coherent alignments. The evaluation was run on a server with 3.46 GHz (6 cores) and 8GB RAM.

2.3 Conference track

The Conference track uses a collection of 16 ontologies from the domain of academic conferences [23]. These ontologies have been created manually by different people and

Table 3: Results for Conference track.

System	RA1 reference			RA2 reference		
	P	F	R	P	F	R
LogMap	0.80	0.68	0.59	0.76	0.63	0.54
LogMap-C	0.82	0.67	0.57	0.78	0.62	0.52
LogMapLt	0.73	0.59	0.50	0.68	0.54	0.45

Table 4: Results for Multifarm track.

System	Different ontologies			Same ontologies		
	P	F	R	P	F	R
LogMap	0.80	0.40	0.28	0.94	0.41	0.27

are of very small size (between 14 and 140 entities). The track uses two reference alignments RA1 and RA2. RA1 contains manually curated mappings between 21 ontology pairs, while RA2 also contains composed mappings based on the alignments in RA1.

Table 3 summarises the average results obtained by the LogMap family. The last column represents the total runtime on generating all 21 alignments. Tests were run on a laptop with Intel Core i5 2.67GHz and 8GB RAM. LogMap ranked 2nd and LogMap-C ranked 3rd. They both produced coherent alignments.

2.4 Multifarm track

This track is based on the translation of the OntoFarm collection of ontologies into 9 different languages [18].

In the OAEI 2014, only LogMap, AML and XMap implemented specific multilingual techniques. Table 4 summarises the results. LogMap achieved very competitive results in terms of precision. Regarding recall, however, there is still room for improvement. In the close future we plan to extend the multilingual module with more sophisticated translation techniques.

2.5 Library track

The library track involves the matching of the STW thesaurus (6,575 classes) and the TheSoz thesaurus (8,376 classes). Both of these thesauri provide vocabulary for economic and social sciences. Table 5 summarises the results obtained by the LogMap family. The track was run on a computer with one 2.4GHz core with 7GB RAM and 2 cores. LogMap ranked 2nd in this track. The results for LogMap* are obtained with a version of the input OWL ontologies using skos labels (i.e. *skos:altLabel* and *skos:prefLabel*).

2.6 Interactive matching track

The interactive track is based on the conference track and it uses the RA1 reference alignment as Oracle. Table 6 summarizes the obtained results by LogMap with the

Table 5: Results for Library track.

System	P	R	F	Time (s)
LogMap*	0.743	0.711	0.681	223
LogMap	0.775	0.705	0.648	74
LogMapLt	0.644	0.703	0.771	9
LogMap-C	0.484	0.342	0.264	22

Table 6: Results for Interactive track.

System	RA1 reference			Avg. Calls	Time (s)
	P	R	F		
LogMap	0.88	0.73	0.64	4	27

Table 7: Summary results for the Large BioMed track

System	Total Time (s)	P	F	R	Inc. Degree.
LogMap	1,751	0.890	0.792	0.719	0.013%
LogMap-Bio	8,634	0.843	0.784	0.744	0.8%
LogMap-C	6,331	0.907	0.688	0.559	0.013%
LogMapLt	317	0.868	0.613	0.532	34.0%

interactive mode activated. LogMap with interactivity improved both the average Precision and Recall wrt LogMap with the interactive mode deactivated (see Section 2.3). LogMap performed on average, 3.91 calls to the Oracle along the 21 matching tasks. LogMap ranked 2nd in the interactive matching track, but it was the system performing less calls to the oracle.

2.7 Large BioMed track

This track consists of finding alignments between the Foundational Model of Anatomy (FMA), SNOMED CT, and the National Cancer Institute Thesaurus (NCI). These ontologies are semantically rich and contain tens of thousands of classes. UMLS Metathesaurus [3] has been selected as the basis for the track reference alignments.

Table 7 summarises the results obtained by the LogMap family. The table shows the total time in seconds to complete all tasks in the track and averages for Precision, Recall, F-measure and Incoherence degree. The track was run on a Ubuntu Laptop with an Intel Core i7-4600U CPU @ 2.10GHz x 4 and allocating 15Gb of RAM..

Only AML and LogMap variants (excluding LogMapLt) generated almost coherent alignments. LogMap ranked 2nd in the track, while LogMap-C and LogMap-Bio obtained the best average Precision and the second best average Recall, respectively. LogMapLt was the fastest to complete all tasks.

Table 8: Results for OA4QA track.

System	Queries	RA1 reference			RAR1 reference		
		P	F	R	P	F	R
LogMap	18/18	0.750	0.741	0.750	0.729	0.728	0.750
LogMapC	18/18	0.722	0.704	0.694	0.722	0.703	0.694
LogMapLt	11/18	0.409	0.379	0.423	0.351	0.348	0.402

Table 9: Results for Instance matching track.

System	Identity		
	P	F	R
LogMap	0.603	0.099	0.054
LogMap-C	0.642	0.078	0.042

2.8 OA4QA track

The Ontology Alignment for Query Answering (OA4QA) track [22] does not follow the classical ontology alignment evaluation with respect to a set of reference alignments. Precision and recall is calculated with respect to the ability of the generated alignments to answer a set of queries in a ontology-based data access scenario where several ontologies exist. Given a query and an ontology pair, a model (or reference) answer set is computed using the correspondent reference alignment for the ontology pair. Precision and recall is calculated with respect to these model answer sets.

In the OAEI 2014 the ontologies and reference alignment (RA1) are based on the conference track. RAR1 is a repaired version of RA1 different from RA2 in the conference track. Table 8 summarises the (average) results for the LogMap family. LogMap and LogMap-C ranked 1st and 2nd in the track, although the number of queries is still not large enough to provide representative values for Precision and Recall. However, the most interesting result is the number of queries a system is able to answer when the computed alignments is incoherent. For example, LogMapLt, since it does not implement mapping repair techniques, is only able to answer 11 of the queries, which damages the obtained precision and recall.

2.9 Instance matching track

The results of LogMap (and LogMap-C) were not as good as previous years. Note that, LogMap does not implement specialised instance matching techniques. Nevertheless, LogMap outperformed two of the participating tools specialised in instance matching. Table 9 summarises the results obtained by LogMap and LogMap-C.

3 General comments and conclusions

3.1 Comments on the results

LogMap, apart from Benchmark and Instance Matching tracks for which does not implement specific techniques, has been one of the top systems in the OAEI 2014. Furthermore, it has also been one of the few systems implementing repair techniques and providing (almost) coherent mappings in all tracks.

LogMap's main weakness relies on the fact that the computation of candidate mappings is based on the similarities between the vocabularies of the input ontologies; hence, there is a direct negative impact in the cases where the ontologies are lexically disparate or do not provide enough lexical information (e.g. Benchmark and Instance Matching).

3.2 Discussions on the way to improve the proposed system

LogMap is now a stable and mature system that has been made available to the community. There are, however, many exciting possibilities for future work. For example we aim at improving the multilingual features and the current use of external resources like BioPortal. Furthremore, we are applying LogMap in practice in the domain of oil and gas industry within the FP7 Optique[8] [16], which presents a very challenging scenario.

3.3 Comments on the OAEI test cases

The number and quality of the OAEI tracks is growing year by year. However, there is always room for improvement:

Comments on the OA4QA track. The new OA4QA track has succesfully shown the negative impact of a incoherent alignment in query answering tasks. However, the number of queries is still small to provide representative values for the F-measure. More queries and more challenging ontologies will make the track more attractive.

Comments on the OAEI interactive matching track. The interactive track has been a very important step forward in the OAEI, however, larger and more challengings tasks should be included. For example, matching tasks (e.g. anatomy and largebio) where the number of questions to the expert user or Oracle may be critical. Furthermore, it is quite unlikely that the expert user will be perfect, thus, the interactive matching track should also consider the evaluation of several Oracles with different error rates such as the evaluation performed in [14].

Comments on the OAEI largebio track. One of the objectives of the largebio track is the creation of a "silver standard" reference alignment by harmonising the output of the different participating systems. In the next OAEI campaign it would be very interesting to actively use this "silver standard" in the construction of the track's reference alignment. This will help to improve the completeness of the reference alignment.

References

1. Agrawal, R., Borgida, A., Jagadish, H.V.: Efficient management of transitive relationships in large data and knowledge bases. In: ACM SIGMOD Conf. on Management of Data. pp. 253–262 (1989)

[8] http://www.optique-project.eu/

2. Baeza-Yates, R.A., Ribeiro-Neto, B.A.: Modern Information Retrieval. ACM Press / Addison-Wesley (1999)
3. Bodenreider, O.: The unified medical language system (UMLS): integrating biomedical terminology. Nucleic Acids Research 32, 267–270 (2004)
4. Chen, X., Xia, W., Jiménez-Ruiz, E., Cross, V.: Extending an ontology alignment system with bioportal: a preliminary analysis. In: Poster at Int'l Sem. Web Conf. (ISWC) (2014)
5. Christophides, V., Plexousakis, D., Scholl, M., Tourtounis, S.: On labeling schemes for the Semantic Web. In: Int'l World Wide Web (WWW) Conf. pp. 544–555 (2003)
6. Cuenca Grau, B., Horrocks, I., Kazakov, Y., Sattler, U.: Modular reuse of ontologies: Theory and practice. J. Artif. Intell. Res. 31, 273–318 (2008)
7. Dowling, W.F., Gallier, J.H.: Linear-time algorithms for testing the satisfiability of propositional Horn formulae. J. Log. Prog. 1(3), 267–284 (1984)
8. Euzenat, J., Rosoiu, M.E., dos Santos, C.T.: Ontology matching benchmarks: Generation, stability, and discriminability. J. Web Sem. 21, 30–48 (2013)
9. Faria, D., Jiménez-Ruiz, E., Pesquita, C., Santos, E., Couto, F.M.: Towards annotating potential incoherences in bioportal mappings. In: 13th Int'l Sem. Web Conf. (ISWC) (2014)
10. Fridman Noy, N., Shah, N.H., Whetzel, P.L., Dai, B., et al.: BioPortal: ontologies and integrated data resources at the click of a mouse. Nucleic Acids Research 37, 170–173 (2009)
11. Ghazvinian, A., Noy, N.F., Jonquet, C., Shah, N.H., Musen, M.A.: What four million mappings can tell you about two hundred ontologies. In: Int'l Sem. Web Conf. (ISWC) (2009)
12. Jiménez-Ruiz, E., Cuenca Grau, B.: LogMap: Logic-based and Scalable Ontology Matching. In: Int'l Sem. Web Conf. (ISWC). pp. 273–288 (2011)
13. Jiménez-Ruiz, E., Cuenca Grau, B., Horrocks, I., Berlanga, R.: Logic-based assessment of the compatibility of UMLS ontology sources. J. Biomed. Sem. 2 (2011)
14. Jiménez-Ruiz, E., Cuenca Grau, B., Zhou, Y., Horrocks, I.: Large-scale interactive ontology matching: Algorithms and implementation. In: Europ. Conf. on Artif. Intell. (ECAI) (2012)
15. Jiménez-Ruiz, E., Meilicke, C., Cuenca Grau, B., Horrocks, I.: Evaluating mapping repair systems with large biomedical ontologies. In: 26th Description Logics Workshop (2013)
16. Kharlamov, E., Jiménez-Ruiz, E., Zheleznyakov, D., et al.: Optique: Towards OBDA Systems for Industry. In: Eur. Sem. Web Conf. (ESWC) Satellite Events. pp. 125–140 (2013)
17. Meilicke, C.: Alignment Incoherence in Ontology Matching. Ph.D. thesis, University of Mannheim (2011)
18. Meilicke, C., Castro, R.G., Freitas, F., van Hage, W.R., Montiel-Ponsoda, E., de Azevedo, R.R., Stuckenschmidt, H., Šváb-Zamazal, O., Svátek, V., Tamilin, A., Trojahn, C., Wang, S.: MultiFarm: a benchmark for multilingual ontology matching. J. Web Sem. (2012)
19. Nebot, V., Berlanga, R.: Efficient retrieval of ontology fragments using an interval labeling scheme. Inf. Sci. 179(24), 4151–4173 (2009)
20. Solimando, A., Jiménez-Ruiz, E., Guerrini, G.: Detecting and correcting conservativity principle violations in ontology-to-ontology mappings. In: Int'l Sem. Web Conf. (ISWC) (2014)
21. Solimando, A., Jiménez-Ruiz, E., Guerrini, G.: A multi-strategy approach for detecting and correcting conservativity principle violations in ontology alignments. In: Proc. of the 11th International Workshop on OWL: Experiences and Directions (OWLED). pp. 13–24 (2014)
22. Solimando, A., Jiménez-Ruiz, E., Pinkel, C.: Evaluating Ontology Alignment Systems in Query Answering Tasks. In: Poster at Int'l Sem. Web Conf. (ISWC) (2014)
23. Šváb, O., Svátek, V., Berka, P., Rak, D., Tomášek, P.: OntoFarm: towards an experimental collection of parallel ontologies. In: Int'l Sem. Web Conf. (ISWC). Poster Session (2005)
24. Zhang, S., Jiménez-Ruiz, E., Cuenca Grau, B.: Inconsistency Repair in Ontology Matching. MSc thesis., University of Oxford (2014), `http://www.cs.ox.ac.uk/isg/projects/LogMap/papers/Master_thesis_Shuo_Zhang.pdf`
25. Zhang, S., Mork, P., Bodenreider, O.: Lessons learned from aligning two representations of anatomy. In: Conf. on Principles of Knowledge Representation and Reasoning (KR) (2004)

Appendix F

ISWC 2014: Conservativity in Ontology Alignments

This appendix reports the paper:

– Alessandro Solimando, Ernesto Jimenez-Ruiz, Giovana Guerrini. Detecting and Correcting Conservativity Principle Violations in Ontology-to-Ontology Mappings. In Proceedings of the International Semantic Web Conference 2014

Detecting and Correcting Conservativity Principle Violations in Ontology-to-Ontology Mappings

Alessandro Solimando[1], Ernesto Jiménez-Ruiz[2], and Giovanna Guerrini[1]

[1] Dipartimento di Informatica, Università di Genova, Italy
[2] Department of Computer Science, University of Oxford, UK

Abstract. In order to enable interoperability between ontology-based systems, ontology matching techniques have been proposed. However, when the generated mappings suffer from logical flaws, their usefulness may be diminished. In this paper we present an approximate method to detect and correct violations to the so-called conservativity principle where novel subsumption entailments between named concepts in one of the input ontologies are considered as unwanted. We show that this is indeed the case in our application domain based on the EU Optique project. Additionally, our extensive evaluation conducted with both the Optique use case and the data sets from the Ontology Alignment Evaluation Initiative (OAEI) suggests that our method is both useful and feasible in practice.

1 Introduction

Ontologies play a key role in the development of the Semantic Web and are being used in many diverse application domains, ranging from biomedicine to energy industry. An application domain may have been modeled with different points of view and purposes. This situation usually leads to the development of different ontologies that intuitively overlap, but they use different naming and modeling conventions.

In particular, this is the case we are facing in the EU Optique project.[3] Optique aims at facilitating scalable end-user access to big data in the oil and gas industry. The project is focused around two demanding use cases provided by *Siemens* and *Statoil*. Optique advocates for an Ontology Based Data Access (OBDA) approach [24] so that end-users formulate queries using the vocabulary of a domain ontology instead of composing queries directly against the database. Ontology-based queries (*e.g.*, SPARQL) are then automatically rewritten to SQL and executed over the database.

In Optique two independently developed ontologies co-exist. The first ontology has been directly bootstrapped from one of the relational databases in Optique and it is linked to the database via *direct ontology-to-database mappings*;[4] while the second ontology is a domain ontology based on the Norwegian Petroleum Directorate (NPD) FactPages[5] [41] and it is currently preferred by Optique end-users to feed the visual query formulation interface[6] [42]. This setting requires the "query formulation" ontology to be linked to the relational database. In Optique we follow two approaches that

[3] `http://www.optique-project.eu/`
[4] `http://www.w3.org/TR/rdb-direct-mapping/`
[5] `http://factpages.npd.no/factpages/`
[6] The query formulation interface has been evaluated with end-users at Statoil.

will complement each other: *(i)* creation of ontology-to-database mappings between the query formulation ontology and the database; *(ii)* creation of *ontology-to-ontology* mappings between the bootstrapped ontology and the query formulation ontology. In this paper we only deal with ontology-to-ontology mappings (or mappings for short). The creation, analysis and evolution of ontology-to-database mappings are also key research topics within Optique, however, they fall out of the scope of this paper.

The problem of (semi-)automatically computing mappings between independently developed ontologies is usually referred to as the *ontology matching problem*. A number of sophisticated ontology matching systems have been developed in the last years [11, 40]. Ontology matching systems, however, rely on lexical and structural heuristics and the integration of the input ontologies and the mappings may lead to many undesired logical consequences. In [19] three principles were proposed to minimize the number of potentially unintended consequences, namely: *(i) consistency principle*, the mappings should not lead to unsatisfiable classes in the integrated ontology, *(ii) locality principle*, the mappings should link entities that have similar *neighbourhoods*, *(iii) conservativity principle*, the mappings should not introduce new semantic relationships between concepts from one of the input ontologies.

The occurrence of these violations is frequent, even in the reference mapping sets of the Ontology Alignment Evaluation Initiative[7] (*OAEI*). Also manually curated alignments, such as *UMLS-Metathesaurus* [3] (*UMLS*), a comprehensive effort for integrating biomedical knowledge bases, suffer from these violations. Violations to these principles may hinder the usefulness of ontology mappings. In particular, in the Optique's scenario, violation of the consistency or conservativity principles will directly affect the quality of the query results, since queries will be rewritten according to the ontology axioms, the ontology-to-ontology mappings and the ontology-to-database mappings.

These principles has been actively investigated in the last years (*e.g.*, [31, 30, 15, 19, 17, 29, 37]). In this paper we focus on the conservativity principle and we explore a variant of violation of this principle which we consider appropriate for the application domain in Optique. Furthermore, this variant of the conservativity principle allows us to reduce the problem to a consistency principle problem. We have implemented a method which relies on the projection of the input ontologies to Horn propositional logic. This projection allows us to be efficient in both the reduction to the consistency principle and the subsequent repair process. Our evaluation suggests that our method is feasible even with the largest test cases of the OAEI campaign.

The remainder of the paper is organised as follows. Section 2 summarises the basics concepts and definitions we will rely on along the paper. In Section 3 we introduce our motivating scenario based on Optique. Section 4 describes our method. In Section 5 we present the conducted evaluation. A comparison with relevant related work is provided in Section 6. Finally, Section 7 gives some conclusions and future work lines.

2 Preliminaries

In this section, we present the formal representation of ontology mappings and the notions of semantic difference, mapping coherence and conservativity principle violation.

[7] `http://oaei.ontologymatching.org/`

2.1 Representation of Ontology Mappings

Mappings are conceptualised as 5-tuples of the form $\langle id, e_1, e_2, n, \rho\rangle$, with id a unique identifier, e_1, e_2 entities in the vocabulary or signature of the relevant input ontologies (*i.e.*, $e_1 \in \mathsf{Sig}(\mathcal{O}_1)$ and $e_2 \in \mathsf{Sig}(\mathcal{O}_2)$), n a confidence measure between 0 and 1, and ρ a relation between e_1 and e_2, typically subsumption, equivalence or disjointness [10].

RDF Alignment [8] is the main format used in the OAEI campaign to represent mappings containing the aforementioned elements. Additionally, mappings are also represented as OWL 2 subclass, equivalence, and disjointness axioms [6]; mapping identifiers (id) and confidence values (n) are then represented as axiom annotations. Such a representation enables the reuse of the extensive range of OWL 2 reasoning infrastructure that is currently available. Note that alternative formal semantics for ontology mappings have been proposed in the literature (*e.g.*, [4]).

2.2 Semantic Consequences of the Integration

The ontology resulting from the integration of two ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$ via a set of mappings $\mathcal{M}$ may entail axioms that do not follow from $\mathcal{O}_1$, $\mathcal{O}_2$, or $\mathcal{M}$ alone. These new semantic consequences can be captured by the notion of *deductive difference* [25].

Intuitively, the deductive difference between $\mathcal{O}$ and $\mathcal{O}'$ w.r.t. a signature Σ (*i.e.*, set of entities) is the set of entailments constructed over Σ that do not hold in $\mathcal{O}$, but do hold in $\mathcal{O}'$. The notion of deductive difference, however, has several drawbacks in practice. First, there is no algorithm for computing the deductive difference in expressive DLs [25]. Second, the number of entailments in the difference can be infinite.

Definition 1 (Approximation of the Deductive Difference). *Let A, B be atomic concepts (including $\top, \bot$), Σ be a signature, $\mathcal{O}$ and $\mathcal{O}'$ be two OWL 2 ontologies. We define the approximation of the Σ-deductive difference between $\mathcal{O}$ and $\mathcal{O}'$ (denoted $\mathsf{diff}^{\approx}_{\Sigma}(\mathcal{O}, \mathcal{O}')$ as the set of axioms of the form $A \sqsubseteq B$ satisfying: (i) $A, B \in \Sigma$, (ii) $\mathcal{O} \not\models A \sqsubseteq B$, and (iii) $\mathcal{O}' \models A \sqsubseteq B$.*

In order to avoid the drawbacks of the deductive difference, in this paper we rely on the *approximation* given in Definition 1. This approximation only requires comparing the classification hierarchies of $\mathcal{O}$ and $\mathcal{O}'$ provided by an OWL 2 reasoner, and it has successfully been used in the past in the context of ontology integration [18].

2.3 Mapping Coherence and Mapping Repair

The consistency principle requires that the vocabulary in $\mathcal{O}_{\mathcal{U}} = \mathcal{O}_1 \cup \mathcal{O}_2 \cup \mathcal{M}$ be satisfiable, assuming the union of input ontologies $\mathcal{O}_1 \cup \mathcal{O}_2$ (without the mappings $\mathcal{M}$) does not contain unsatisfiable concepts. Thus $\mathsf{diff}^{\approx}_{\Sigma}(\mathcal{O}_1 \cup \mathcal{O}_2, \mathcal{O}_{\mathcal{U}})$ should not contain any axiom of the form $A \sqsubseteq \bot$, for any $A \in \Sigma = \mathsf{Sig}(\mathcal{O}_1 \cup \mathcal{O}_2)$.

Definition 2 (Mapping Incoherence). *A set of mappings $\mathcal{M}$ is incoherent with respect to $\mathcal{O}_1$ and $\mathcal{O}_2$, if there exists a class A, in the signature of $\mathcal{O}_1 \cup \mathcal{O}_2$, such that $\mathcal{O}_1 \cup \mathcal{O}_2 \not\models A \sqsubseteq \bot$ and $\mathcal{O}_1 \cup \mathcal{O}_2 \cup \mathcal{M} \models A \sqsubseteq \bot$.*

An incoherent set of mappings $\mathcal{M}$ can be fixed by removing mappings from $\mathcal{M}$. This process is referred to as *mapping repair* (or repair for short).

Definition 3 (Mapping Repair). *Let $\mathcal{M}$ be an incoherent set of mappings w.r.t. $\mathcal{O}_1$ and $\mathcal{O}_2$. A set of mappings $\mathcal{R} \subseteq \mathcal{M}$ is a mapping repair for $\mathcal{M}$ w.r.t. $\mathcal{O}_1$ and $\mathcal{O}_2$ iff $\mathcal{M} \setminus \mathcal{R}$ is coherent w.r.t. $\mathcal{O}_1$ and $\mathcal{O}_2$.*

A trivial repair is $\mathcal{R} = \mathcal{M}$, since an empty set of mappings is trivially coherent (according to Definition 2). Nevertheless, the objective is to remove as few mappings as possible. Minimal (mapping) repairs are typically referred to in the literature as *mapping diagnoses* [29] — a term coined by Reiter [36] and introduced to the field of ontology debugging in [39]. A repair or diagnosis can be computed by extracting the justifications for the unsatisfiable concepts (*e.g.*, [38, 22, 43]), and selecting a hitting set of mappings to be removed, following a minimality criteria (*e.g.*, the number of removed mappings). However, justification-based technologies do not scale when the number of unsatisfiabilities is large (a typical scenario in mapping repair problems [16]). To address this scalability issue, mapping repair systems usually compute an *approximate repair* using incomplete reasoning techniques (*e.g.*, [17, 29, 37]). An approximate repair $\mathcal{R}^{\approx}$ does not guarantee that $\mathcal{M} \setminus \mathcal{R}^{\approx}$ is coherent, but it will (in general) significantly reduce the number of unsatisfiabilities caused by the original set of mappings $\mathcal{M}$.

2.4 Conservativity Principle

The conservativity principle (general notion) states that the integrated ontology $\mathcal{O}_{\mathcal{U}} = \mathcal{O}_1 \cup \mathcal{O}_2 \cup \mathcal{M}$ should not induce any change in the concept hierarchies of the input ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$. That is, the sets $\mathsf{diff}^{\approx}_{\Sigma_1}(\mathcal{O}_1, \mathcal{O}_{\mathcal{U}})$ and $\mathsf{diff}^{\approx}_{\Sigma_2}(\mathcal{O}_2, \mathcal{O}_{\mathcal{U}})$ must be empty for signatures $\Sigma_1 = \mathsf{Sig}(\mathcal{O}_1)$ and $\Sigma_2 = \mathsf{Sig}(\mathcal{O}_2)$, respectively.

In [19] a lighter variant of the conservativity principle was proposed. This variant required that the mappings $\mathcal{M}$ alone should not introduce new subsumption relationships between concepts from one of the input ontologies. That is, the set $\mathsf{diff}^{\approx}_{\Sigma}(\mathcal{O}_1, \mathcal{O}_1 \cup \mathcal{M})$ (resp. $\mathsf{diff}^{\approx}_{\Sigma}(\mathcal{O}_2, \mathcal{O}_2 \cup \mathcal{M})$) must be empty for $\Sigma = \mathsf{Sig}(\mathcal{O}_1)$ (resp. $\Sigma = \mathsf{Sig}(\mathcal{O}_2)$).

In this paper we propose a different variant of the conservativity principle where we require that the integrated ontology $\mathcal{O}_{\mathcal{U}}$ does not introduce new subsumption relationships between concepts from one of the input ontologies, unless they were already involved in a subsumption relationship or they shared a common descendant. Note that we assume that the mappings $\mathcal{M}$ are coherent with respect to $\mathcal{O}_1$ and $\mathcal{O}_2$.

Definition 4 (Conservativity Principle Violations). *Let A, B, C be atomic concepts (not including $\top, \bot$), let $\mathcal{O}$ be one of the input ontologies, let $\mathsf{Sig}(\mathcal{O})$ be its signature, and let $\mathcal{O}_{\mathcal{U}}$ be the integrated ontology. We define the set of conservativity principle violations of $\mathcal{O}_{\mathcal{U}}$ w.r.t. $\mathcal{O}$ (denoted $\mathsf{consViol}(\mathcal{O}, \mathcal{O}_{\mathcal{U}})$) as the set of axioms of the form $A \sqsubseteq B$ satisfying: (i) $A, B, C \in \mathsf{Sig}(\mathcal{O})$, (ii) $A \sqsubseteq B \in \mathsf{diff}^{\approx}_{\mathsf{Sig}(\mathcal{O})}(\mathcal{O}, \mathcal{O}_{\mathcal{U}})$, (iii) $\mathcal{O} \not\models B \sqsubseteq A$, and (iv) there is no C s.t. $\mathcal{O} \models C \sqsubseteq A$, and $\mathcal{O} \models C \sqsubseteq B$.*

This variant of the conservativity principle follows the *assumption of disjointness* proposed in [38]. That is, if two atomic concepts A, B from one of the input ontologies are not involved in a subsumption relationship nor share a common subconcept (excluding $\bot$) they can be considered as disjoint. Hence, the conservativity principle can be reduced to the consistency principle, if the input ontologies are extended with sufficient disjointness axioms. This reduction will allow us to reuse the available infrastructure and techniques for mapping repair.

Table 1. Fragments of the ontologies used in Optique.

Ontology $\mathcal{O}_1$	**Ontology $\mathcal{O}_2$**
α_1 WellBore $\sqsubseteq$ $\exists$belongsTo.Well	β_1 Exploration_well $\sqsubseteq$ Well
α_2 WellBore $\sqsubseteq$ $\exists$hasOperator.Operator	β_2 Explor_borehole $\sqsubseteq$ Borehole
α_3 WellBore $\sqsubseteq$ $\exists$locatedIn.Field	β_3 Appraisal_exp_borehole $\sqsubseteq$ Explor_borehole
α_4 AppraisalWellBore $\sqsubseteq$ WellBore	β_4 Appraisal_well $\sqsubseteq$ Well
α_5 ExplorationWellBore $\sqsubseteq$ WellBore	β_5 Field $\sqsubseteq$ $\exists$hasFieldOperator.Field_operator
α_6 Operator $\sqsubseteq$ Owner	β_6 Field_operator $\sqcap$ Owner $\sqsubseteq$ Field_owner
α_7 Operator $\sqsubseteq$ Company	β_7 Company $\sqsubseteq$ Field_operator
α_8 Field $\sqsubseteq$ $\exists$hasOperator.Company	β_8 Field_owner $\sqsubseteq$ Owner
α_9 Field $\sqsubseteq$ $\exists$hasOwner.Owner	β_9 Borehole $\sqsubseteq$ Continuant $\sqcup$ Occurrent

Table 2. Ontology mappings for the vocabulary in $\mathcal{O}_1$ and $\mathcal{O}_2$.

Mappings $\mathcal{M}$				
id	**e₁**	**e₂**	**n**	**ρ**
m_1	$\mathcal{O}_1$:Well	$\mathcal{O}_2$:Well	0.9	$\equiv$
m_2	$\mathcal{O}_1$:WellBore	$\mathcal{O}_2$:Borehole	0.7	$\equiv$
m_3	$\mathcal{O}_1$:ExplorationWellBore	$\mathcal{O}_2$:Exploration_well	0.6	$\sqsubseteq$
m_4	$\mathcal{O}_1$:ExplorationWellBore	$\mathcal{O}_2$:Explor_borehole	0.8	$\equiv$
m_5	$\mathcal{O}_1$:AppraisalWellBore	$\mathcal{O}_2$:Appraisal_exp_borehole	0.7	$\equiv$
m_6	$\mathcal{O}_1$:Field	$\mathcal{O}_2$:Field	0.9	$\equiv$
m_7	$\mathcal{O}_1$:Operator	$\mathcal{O}_2$:Field_operator	0.7	$\sqsupseteq$
m_8	$\mathcal{O}_1$:Company	$\mathcal{O}_2$:Company	0.9	$\equiv$
m_9	$\mathcal{O}_1$:hasOperator	$\mathcal{O}_2$:hasFieldOperator	0.6	$\equiv$
m_{10}	$\mathcal{O}_1$:Owner	$\mathcal{O}_2$:Owner	0.9	$\equiv$

3 Conservativity Principle Violations in Practice

In this section, we show the problems led by the violation of the conservativity principle when integrating ontologies via mappings in a real-world scenario. To this end, we consider as motivating example a use case based on the Optique's application domain.

Table 1 shows the fragments of two ontologies in the context of the oil and gas industry. The ontology $\mathcal{O}_1$ has been directly bootstrapped from a relational database in Optique, and it is linked to the data via direct ontology-to-database mappings. The ontology $\mathcal{O}_2$, instead, is a domain ontology, based on the NPD FactPages, preferred by Optique end-users to feed the visual query formulation interface.[8]

The integration via ontology matching of $\mathcal{O}_1$ and $\mathcal{O}_2$ is required since the vocabulary in $\mathcal{O}_2$ is used to formulate queries, but only the vocabulary of $\mathcal{O}_1$ is connected to the database.[9] Consider the set of mappings $\mathcal{M}$ in Table 2 between $\mathcal{O}_1$ and $\mathcal{O}_2$ generated by an off-the-shelf ontology alignment system. As described in Section 2.1, mappings are represented as 5-tuples; for example the mapping m_2 suggests an equivalence relationship between the entities $\mathcal{O}_1$:WellBore and $\mathcal{O}_2$:Borehole, with confidence 0.7.

The integrated ontology $\mathcal{O}_{\mathcal{U}} = \mathcal{O}_1 \cup \mathcal{O}_2 \cup \mathcal{M}$, however, violates the conservativity principle, according to Definition 4, and introduces non desired subsumption relationships (see Table 3). Note that the entailments σ_4 and σ_5 are not included in our variant of conservativity violation, since $\mathcal{O}_1$:Company and $\mathcal{O}_1$:Operator (resp. $\mathcal{O}_2$:Field_operator and $\mathcal{O}_2$:Company) are involved in a subsumption relationship in $\mathcal{O}_1$ (resp. $\mathcal{O}_2$). How-

[8] In Optique we use OWL 2 QL ontologies for query rewriting, while the query formulation may be based on much richer OWL 2 ontologies. The axioms that fall outside the OWL 2 QL profile are either approximated or not considered for the rewriting.

[9] As mentioned in Section 1, in this paper we only focus on ontology-to-ontology mappings.

Table 3. Example of conservativity principle violations.

σ	**Entailment:**	**follows from:**	**Violation?**
σ_1	$\mathcal{O}_2$:Explor_borehole $\sqsubseteq$ $\mathcal{O}_2$:Exploration_well	m_3, m_4	YES
σ_2	$\mathcal{O}_1$:AppraisalWellBore $\sqsubseteq$ $\mathcal{O}_1$:ExplorationWellBore	β_3, m_4, m_5	YES
σ_3	$\mathcal{O}_2$:Field_operator $\sqsubseteq$ $\mathcal{O}_2$:Field_owner	$\alpha_6, \beta_6, m_7, m_{10}$	YES
σ_4	$\mathcal{O}_1$:Company $\equiv$ $\mathcal{O}_1$:Operator	$\alpha_7, \beta_7, m_7, m_8$	NO (*)
σ_5	$\mathcal{O}_2$:Field_operator $\equiv$ $\mathcal{O}_2$:Company		
σ_6	$\mathcal{O}_1$:Company $\sqsubseteq$ $\mathcal{O}_1$:Owner	σ_4, α_6	YES
σ_7	$\mathcal{O}_2$:Company $\sqsubseteq$ $\mathcal{O}_2$:Field_owner	σ_3, σ_5	YES

ever, these entailments lead to other violations included in our variant (σ_6 and σ_7), and may also be considered as violations. These conservativity principle violations may hinder the usefulness of the generated ontology mappings since may affect the quality of the results when performing OBDA queries over the vocabulary of $\mathcal{O}_2$.

Example 1. Consider the following conjunctive query $\mathsf{CQ}(\mathsf{x}) \leftarrow \mathcal{O}_2{:}\mathsf{Well}(\mathsf{x})$. The query asks for wells and has been formulated from the Optique's query formulation interface, using the vocabulary of $\mathcal{O}_2$. The query is rewritten, according to the ontology axioms and mappings $\beta_1, \beta_4, m_1, m_3, m_4$ in $\mathcal{O}_{\mathcal{U}} = \mathcal{O}_1 \cup \mathcal{O}_2 \cup \mathcal{M}$, into the following union of conjunctive queries $\mathsf{UCQ}(\mathsf{x}) \leftarrow \mathcal{O}_2{:}\mathsf{Well}(\mathsf{x}) \cup \mathcal{O}_1{:}\mathsf{Well}(\mathsf{x}) \cup \mathcal{O}_2{:}\mathsf{Exploration\_well}(\mathsf{x}) \cup \mathcal{O}_2{:}\mathsf{Appraisal\_well}(\mathsf{x}) \cup \mathcal{O}_1{:}\mathsf{ExplorationWellBore}(\mathsf{x}) \cup \mathcal{O}_2{:}\mathsf{Explor\_borehole}(\mathsf{x})$. Since only the vocabulary of $\mathcal{O}_1$ is linked to the data, the union of conjunctive queries could be simplified as $\mathsf{UCQ}(\mathsf{x}) \leftarrow \mathsf{Well}(\mathsf{x}) \cup \mathsf{ExplorationWellBore}(\mathsf{x})$, which will clearly lead to non desired results. The original query was only asking for wells, while the rewritten query will also return data about exploration wellbores.

We have shown that the quality of the mappings in terms of conservativity principle violations will directly affect the quality of the query results. Therefore, the detection and repair of these violations arise as an important quality assessment step in Optique.

4 Methods

We have reduced the problem of detecting and solving conservativity principle violations, following our notion of conservativity (see Section 2), to a mapping (incoherence) repair problem. Currently, our method relies on the indexing and reasoning techniques implemented in *LogMap*, an ontology matching and mapping repair system [17, 20, 21].

Algorithm 1 shows the pseudocode of the implemented method. The algorithm accepts as input two OWL 2 ontologies, $\mathcal{O}_1$ and $\mathcal{O}_2$, and a set of mappings $\mathcal{M}$ which are coherent[10] with respect to $\mathcal{O}_1$ and $\mathcal{O}_2$. Additionally, an optimised variant to add disjointness axioms can be selected. The algorithm outputs the number of added disjointness during the process $disj$, a set of mappings $\mathcal{M}'$, and an (approximate) repair $\mathcal{R}^{\approx}$ such that $\mathcal{M}' = \mathcal{M} \setminus \mathcal{R}^{\approx}$. The (approximate) repair $\mathcal{R}^{\approx}$ aims at solving most of the conservativity principle violations of $\mathcal{M}$ with respect to $\mathcal{O}_1$ and $\mathcal{O}_2$. We next describe the techniques used in each step.

[10] Note that $\mathcal{M}$ may be the result of a prior mapping (incoherence) repair process.

Algorithm 1 Algorithm to detect and solve conservativity principle violations

```
Input: $\mathcal{O}_1, \mathcal{O}_2$: input ontologies; $\mathcal{M}$: (coherent) input mappings; $Optimization$: Boolean value
Output: $\mathcal{M}'$: output mappings; $\mathcal{R}^{\approx}$: approximate repair; $disj$: number of disjointness rules
 1: $\langle \mathcal{O}'_1, \mathcal{O}'_2 \rangle :=$ ModuleExtractor$(\mathcal{O}_1, \mathcal{O}_2, \mathcal{M})$
 2: $\langle \mathcal{P}_1, \mathcal{P}_2 \rangle :=$ PropositionalEncoding$(\mathcal{O}'_1, \mathcal{O}'_2)$
 3: $SI_1 :=$ StructuralIndex$(\mathcal{O}'_1)$
 4: $SI_2 :=$ StructuralIndex$(\mathcal{O}'_2)$
 5: if ($Optimization =$ true) then
 6:     $SI_{\mathcal{U}} :=$ StructuralIndex$(\mathcal{O}'_1 \cup \mathcal{O}'_2 \cup \mathcal{M})$
 7:     $\langle \mathcal{P}^d_1, disj_1 \rangle :=$ DisjointAxiomsExtensionOptimized$(\mathcal{P}_1, SI_1, SI_{\mathcal{U}})$      ▷ See Algorithm 3
 8:     $\langle \mathcal{P}^d_2, disj_2 \rangle :=$ DisjointAxiomsExtensionOptimized$(\mathcal{P}_2, SI_2, SI_{\mathcal{U}})$
 9: else
10:     $\langle \mathcal{P}^d_1, disj_1 \rangle :=$ DisjointAxiomsExtensionBasic$(\mathcal{P}_1, SI_1)$      ▷ See Algorithm 2
11:     $\langle \mathcal{P}^d_2, disj_2 \rangle :=$ DisjointAxiomsExtensionBasic$(\mathcal{P}_2, SI_2)$
12: end if
13: $\langle \mathcal{M}', \mathcal{R}^{\approx} \rangle :=$ MappingRepair$(\mathcal{P}^d_1, \mathcal{P}^d_2, \mathcal{M})$      ▷ See Algorithm 2 in [21]
14: $disj := disj_1 + disj_2$
15: return $\langle \mathcal{M}', \mathcal{R}^{\approx}, disj \rangle$
```

Module Extraction. In order to reduce the size of the problem our method extracts two locality-based modules [7], one for each input ontology, using the entities involved in the mappings $\mathcal{M}$ as seed signatures for the module extractor (step 1 in Algorithm 1). These modules preserve the semantics for the given entities, can be efficiently computed, and are typically much smaller than the original ontologies.

Propositional Horn Encoding. The modules $\mathcal{O}'_1$ and $\mathcal{O}'_2$ are encoded as the Horn propositional theories, $\mathcal{P}_1$ and $\mathcal{P}_2$ (step 2 in Algorithm 1). This encoding includes rules of the form $A_1 \wedge \ldots \wedge A_n \rightarrow B$. For example, the concept hierarchy provided by an OWL 2 reasoner (*e.g.*, [32, 23]) will be encoded as $A \rightarrow B$ rules, while the explicit disjointness relationships between concepts will be represented as $A_i \wedge A_j \rightarrow \mathsf{false}$. Note that the input mappings $\mathcal{M}$ can already be seen as propositional implications. This encoding is key to the mapping repair process.

Example 2. Consider the ontologies and mappings in Tables 1 and 2. The axiom β_6 is encoded as $\mathsf{Field\_operator} \wedge \mathsf{Owner} \rightarrow \mathsf{Field\_owner}$, while the mapping m_2 is translated into rules $\mathcal{O}_1$:WellBore $\rightarrow$ $\mathcal{O}_2$:Borehole, and $\mathcal{O}_2$:Borehole $\rightarrow$ $\mathcal{O}_1$:WellBore.

Structural Index. The concept hierarchies provided by an OWL 2 reasoner (excluding $\bot$) and the explicit disjointness axioms of the modules $\mathcal{O}'_1$ and $\mathcal{O}'_2$ are efficiently indexed using an interval labelling schema [1] (steps 3 and 4 in Algorithm 1). This structural index exploits an optimised data structure for storing directed acyclic graphs (DAGs), and allows us to answer many entailment queries over the concept hierarchy as an index lookup operation, and hence without the need of an OWL 2 reasoner. This kind of index has shown to significantly reduce the cost of answering taxonomic queries [5, 33] and disjointness relationships queries [17, 20].

Disjointness Axioms Extension. In order to reduce the conservativity problem to a mapping incoherence repair problem following the notion of *assumption of disjointness*, we need to automatically add sufficient disjointness axioms into each module $\mathcal{O}'_i$. However, the insertion of additional disjointness axioms δ may lead to unsatisfiable classes in $\mathcal{O}'_i \cup \delta$.

Algorithm 2 Basic disjointness axioms extension

```
Input: $\mathcal{P}$: propositional theory; $SI$: structural index
Output: $\mathcal{P}^d$: extended propositional theory;$disj$: number of disjointness rules
 1: $disj := 0$
 2: $\mathcal{P}^d := \mathcal{P}$
 3: for each pair $\langle A, B\rangle \in$ OrderedVariablePairs($\mathcal{P}$) do
 4:     if not (areDisj($SI, A, B$) or inSubSupRel($SI, A, B$) or shareDesc($SI, A, B$)) then
 5:         $\mathcal{P}^d := \mathcal{P}^d \cup \{A \wedge B \rightarrow false\}$
 6:         $SI :=$ updateIndex($SI, A \sqcap B \rightarrow \bot$)
 7:         $disj := disj + 1$
 8:     end if
 9: end for
10: return $\langle \mathcal{P}^d, disj\rangle$
```

Example 3. Consider the axiom β_9 from Table 1. Following the *assumption of disjointness* a very naïve algorithm would add disjointness axioms between Borehole, Continuant and Occurrent, which would make Borehole unsatisfiable.

In order to detect if each candidate disjointness axiom leads to an unsatisfiability, a non naïve algorithm requires to make an extensive use of an OWL 2 reasoner. In large ontologies, however, such extensive use of the reasoner may be prohibitive. Our method, in order to address this issue, exploits the propositional encoding and structural index of the input ontologies. Thus, checking if $\mathcal{O}'_i \cup \delta$ contains unsatisfiable classes is restricted to the Horn propositional case.

We have implemented two algorithms to extend the propositional theories $\mathcal{P}_1$ and $\mathcal{P}_2$ with disjointness rules of the form $A \wedge B \rightarrow \bot$ (see steps 5-12 in Algorithm 1). These algorithms guarantee that, for every propositional variable A in the extended propositional theory $\mathcal{P}^d_i$ (with $i \in \{1, 2\}$), the theory $\mathcal{P}^d_i \cup \{true \rightarrow A\}$ is satisfiable. Note that this does not necessarily hold if the disjointness axioms are added to the OWL 2 ontology modules, $\mathcal{O}'_1$ and $\mathcal{O}'_2$, as discussed above.

Algorithm 2 presents a (basic) algorithm to add as many disjointness rules as possible, for every pair of propositional variables A, B in the propositional theory $\mathcal{P}$ given as input. In order to minimize the number of necessary disjointness rules, the variables in $\mathcal{P}$ are ordered in pairs following a top-down approach. The algorithm exploits the structural index SI to check if two propositional variables (*i.e.*, classes in the input ontologies) are disjoint (areDisj(SI, A, B)), they keep a sub/super-class relationship (inSubSupRel(SI, A, B)), or they share a common descendant (shareDesc(SI, A, B)) (step 4 in Algorithm 2). Note that the structural index is also updated to take into account the new disjointness rules (step 6 in Algorithm 2).

The addition of disjointness rules in Algorithm 2, however, may be prohibitive for large ontologies (see Section 5). Intuitively, in order to reduce the number of disjointness axioms, one should only focus on the cases where a conservativity principle violation occurs in the integrated ontology $\mathcal{O}_{\mathcal{U}} = \mathcal{O}'_1 \cup \mathcal{O}'_2 \cup \mathcal{M}$, with respect to one of the ontology modules $\mathcal{O}'_i$ (with $i \in \{1, 2\}$); *i.e.*, adding a disjointness axiom between each pair of classes $A, B \in \mathcal{O}'_i$ such that $A \sqsubseteq B \in$ consViol($\mathcal{O}'_i, \mathcal{O}_{\mathcal{U}}$), as in Definition 4. Algorithm 3 implements this idea for the Horn propositional case and extensively exploits the structural indexing to identify the conservativity principle violations (step 3 in Algorithm 3). Note that this algorithm also requires to compute the structural index

Algorithm 3 Optimised disjointness axioms extension

```
Input: $\mathcal{P}$: propositional theory; $SI$: structural index $SI_{\mathcal{U}}$: structural index of the union ontology
Output: $\mathcal{P}^d$: extended propositional theory;$disj$: number of disjointness rules
 1: $disj := 0$
 2: $\mathcal{P}^d := \mathcal{P}$
 3: for $A \to B \in$ ConservativityViolations$(SI, SI_{\mathcal{U}})$ do
 4:     if not (areDisj$(SI, A, B)$) then
 5:         $\mathcal{P}^d := \mathcal{P}^d \cup \{A \wedge B \to false\}$
 6:         $SI :=$ updateIndex$(SI, A \sqcap B \to \bot)$
 7:         $disj := disj + 1$
 8:     end if
 9: end for
10: return $\langle \mathcal{P}^d, disj \rangle$
```

of the integrated ontology, and thus its classification with an OWL 2 reasoner (step 6 in Algorithm 1). The classification of the integrated ontology is known to be typically much higher than the classification of the input ontologies individually [16]. However, this was not a bottleneck in our experiments, as shown in Section 5.

Mapping Repair. The step 13 of Algorithm 1 uses the mapping (incoherence) repair algorithm presented in [17, 21] for the extended Horn propositional theories $\mathcal{P}_1^d$ and $\mathcal{P}_2^d$, and the input mappings $\mathcal{M}$. The mapping repair process exploits the Dowling-Gallier algorithm for propositional Horn satisfiability [9] and checks, for every propositional variable $A \in \mathcal{P}_1^d \cup \mathcal{P}_2^d$, the satisfiability of the propositional theory $\mathcal{P}_A = \mathcal{P}_1^d \cup \mathcal{P}_2^d \cup \mathcal{M} \cup \{true \to A\}$. Satisfiability of $\mathcal{P}_A$ is checked in worst-case linear time in the size of $\mathcal{P}_A$, and the number of Dowling-Gallier calls is also linear in the number of propositional variables in $\mathcal{P}_1^d \cup \mathcal{P}_2^d$. In case of unsatisfiability, the algorithm also allows us to record *conflicting* mappings involved in the unsatisfiability, which will be considered for the subsequent repair process. The unsatisfiability will be fixed by removing some of the identified mappings. In case of multiple options, the mapping confidence will be used as a differentiating factor.[11]

Example 4. Consider the propositional encoding $\mathcal{P}_1$ and $\mathcal{P}_2$ of the axioms of Table 1 and the mappings $\mathcal{M}$ in Table 2, seen as propositional rules. $\mathcal{P}_1^d$ and $\mathcal{P}_2^d$ have been created by adding disjointness rules to $\mathcal{P}_1$ and $\mathcal{P}_2$, according to Algorithm 2 or 3. For example, $\mathcal{P}_2^d$ includes the rule $\psi = \mathcal{O}_2$:Well $\wedge$ $\mathcal{O}_2$:Borehole $\to false$. The mapping repair algorithm identifies the propositional theory $\mathcal{P}_1^d \cup \mathcal{P}_2^d \cup \mathcal{M} \cup \{true \to \mathcal{O}_1$:ExplorationWellbore$\}$ as unsatisfiable. This is due to the combination of the mappings m_3 and m_4, the propositional projection of axioms β_1 and β_2, and the rule ψ. The mapping repair algorithm also identifies m_3 and m_4 as the cause of the unsatisfiability, and discards m_3, since its confidence is smaller than that of m_4 (see Table 2).

Algorithm 1 gives as output the number of added disjointness rules during the process $disj$, a set of mappings $\mathcal{M}'$, and an (approximate) repair $\mathcal{R}^{\approx}$ such that $\mathcal{M}' = \mathcal{M} \setminus \mathcal{R}^{\approx}$. $\mathcal{M}'$ is coherent with respect to $\mathcal{P}_1^d$ and $\mathcal{P}_2^d$ (according to the propositional case of Definition 2). Furthermore, the propositional theory $\mathcal{P}_1 \cup \mathcal{P}_2 \cup \mathcal{M}'$ does not

[11] In scenarios where the confidence of the mapping is missing (*e.g.*, in reference or manually created mapping sets) or unreliable, our mapping repair technique computes fresh confidence values based on the locality principle [19].

Algorithm 4 Conducted evaluation over the Optique and OAEI data sets

```
Input: $\mathcal{O}_1$, $\mathcal{O}_2$: input ontologies $\mathcal{M}$: reference mappings for $\mathcal{O}_1$ and $\mathcal{O}_2$
1: $\mathcal{O}_{\mathcal{U}} := \mathcal{O}_1 \cup \mathcal{O}_2 \cup \mathcal{M}$
2: Store size of $\mathsf{Sig}(\mathcal{O}_1)$ (I), $\mathsf{Sig}(\mathcal{O}_2)$ (II) and $\mathcal{M}$ (III)
3: Compute number of conservativity principle violations (our variant as in Definition 4):
   $\mathsf{consViol} := |\mathsf{consViol}(\mathcal{O}_1, \mathcal{O}_{\mathcal{U}})| + |\mathsf{consViol}(\mathcal{O}_2, \mathcal{O}_{\mathcal{U}})|$ (IV)
4: Compute number of conservativity principle violations (general notion as in Section 2.4):
   $\mathsf{diff}^{\approx} := |\mathsf{diff}^{\approx}_{\mathsf{Sig}(\mathcal{O}_1)}(\mathcal{O}_1, \mathcal{O}_{\mathcal{U}})| + |\mathsf{diff}^{\approx}_{\mathsf{Sig}(\mathcal{O}_2)}(\mathcal{O}_2, \mathcal{O}_{\mathcal{U}})|$ (V)
5: Compute two repairs $\mathcal{R}^{\approx}$ using Algorithm 1 for $\mathcal{O}_1$, $\mathcal{O}_2$, $\mathcal{M}$, with the $Optimization$ set to false (see Table 5) and
   true (see Table 6)
6: Store number of added disjointness $disj$ (VI and XII), size of repair $|\mathcal{R}^{\approx}|$ (VII and XIII), time to compute disjointness
   rules $\mathrm{t}_d$ (VIII and XIV), and time to compute the mapping repair $\mathrm{t}_r$ (IX and XV)
7: $\mathcal{O}_{\mathcal{U}} := \mathcal{O}_1 \cup \mathcal{O}_2 \cup \mathcal{M} \setminus \mathcal{R}^{\approx}$
8: Compute number of remaining conservativity principle violations (our variant):
   $\mathsf{consViol} := |\mathsf{consViol}(\mathcal{O}_1, \mathcal{O}_{\mathcal{U}})| + |\mathsf{consViol}(\mathcal{O}_2, \mathcal{O}_{\mathcal{U}})|$ (X and XVI)
9: Compute number of remaining conservativity principle violations (general notion):
   $\mathsf{diff}^{\approx} := |\mathsf{diff}^{\approx}_{\mathsf{Sig}(\mathcal{O}_1)}(\mathcal{O}_1, \mathcal{O}_{\mathcal{U}})| + |\mathsf{diff}^{\approx}_{\mathsf{Sig}(\mathcal{O}_2)}(\mathcal{O}_2, \mathcal{O}_{\mathcal{U}})|$ (XI and XVII)
```

contain any conservativity principle violation with respect to $\mathcal{P}_1$ and $\mathcal{P}_2$ (according to the propositional case of Definition 4). However, our encoding is incomplete, and we cannot guarantee that $\mathcal{O}'_1 \cup \mathcal{O}'_2 \cup \mathcal{M}'$ does not contain conservativity principle violations with respect to $\mathcal{O}'_1$ and $\mathcal{O}'_2$. Nonetheless, our evaluation suggests that the number of remaining violations after repair is typically small (see Section 5).

5 Evaluation

In this section we evaluate the feasibility of using our method to detect and correct conservativity principle violations in practice. To this end we have conducted the evaluation in Algorithm 4 (the Roman numbers refer to stored measurements) over the Optique's use case and the ontologies and reference mapping sets of the OAEI 2013 campaign:[12]

i *Optique*'s use case is based on the NPD ontology and a bootstrapped ontology (BootsOnto) from one of the Optique databases. The mappings between these ontologies were semi-automatically created using the ontology matcher *LogMap* [20]. Although the NPD ontology is small with respect to the size of the bootstrapped ontology, its vocabulary covers a large portion of the current query catalog in Optique.
ii *LargeBio*: this dataset includes the biomedical ontologies FMA, NCI and (a fragment of) SNOMED, and reference mappings based on the UMLS [3].
iii *Anatomy*: the Anatomy dataset involves the Adult Mouse Anatomy (MO) ontology and a fragment of the NCI ontology ($\mathrm{NCI}_{\mathrm{Anat}}$), describing human anatomy. The reference alignment has been manually curated [48].
iv *Library*: this OAEI dataset includes the real-word thesauri STW and TheSoz from the social sciences. The reference mappings have been manually validated.
v *Conference*: this dataset uses a collection of 16 ontologies from the domain of academic conferences [46]. Currently, there are 21 manually created mapping sets among 7 of the ontologies.

[12] Note that the reference mappings of the OAEI 2013 campaign are coherent with respect to the test case ontologies [13]. More information about the used ontology versions can be found in `http://oaei.ontologymatching.org/2013/`

Table 4. Test cases and violations with original reference mappings. BootsOnto contains around 3,000 concepts, and a large number of properties.

Dataset	$\mathcal{O}_1 \sim \mathcal{O}_2$	Problem size			Original Violations	
		I	**II**	**III**	**IV**	**V**
		$\lvert\mathsf{Sig}(\mathcal{O}_1)\rvert$	$\lvert\mathsf{Sig}(\mathcal{O}_2)\rvert$	$\lvert\mathcal{M}\rvert$	consViol	$\mathsf{diff}^{\approx}$
Optique	**NPD∼BootsOnto**	757	40,671	102	214	220
LargeBio	**SNOMED∼NCI**	122,519	66,914	36,405	>525,515	>546,181
	FMA∼SNOMED	79,042	122,519	17,212	125,232	127,668
	FMA∼NCI	79,042	66,914	5,821	19,740	19,799
Anatomy	**MO∼NCI$_{\text{Anat}}$**	2,747	3,306	3,032	1,321	1,335
Library	**STW∼TheSoz**	6,575	8,376	6,322	42,045	42,872
Conference	**cmt∼confof**	89	75	32	11	11
	conference∼edas	124	154	34	8	8
	conference∼iasted	124	182	28	9	9
	confof∼ekaw	75	107	40	6	6
	edas∼iasted	154	182	38	7	7

Table 5. Results of our basic method to detect and solve conservativity principle violations.

Dataset	$\mathcal{O}_1 \sim \mathcal{O}_2$	Solution size		Times		Remaining Violations	
		VI	**VII**	**VIII**	**IX**	**X**	**XI**
		#disj	$\lvert\mathcal{R}^{\approx}\rvert$	t_d(s)	t_r(s)	consViol	$\mathsf{diff}^{\approx}$
Optique	**NPD∼BootsOnto**	4,716,685	49	9,840	121	0	0
LargeBio	**SNOMED∼NCI**	–	–	–	–	–	–
	FMA∼SNOMED	1,106,259	8,234	35,817	1,127	0	121
	FMA∼NCI	347,801	2,176	2,471	38	103	112
Anatomy	**MO∼NCI$_{\text{Anat}}$**	1,331,374	461	397	56	0	3
Library	**STW∼TheSoz**	591,115	2,969	4,126	416	0	24
Conference	**cmt∼confof**	50	6	0.01	0.01	0	0
	conference∼edas	774	6	0.03	0.01	0	0
	conference∼iasted	2,189	4	0.06	0.02	0	0
	confof∼ekaw	296	6	0.02	0.01	0	0
	edas∼iasted	1,210	4	0.06	0.02	1	1

Table 4 shows the size of the evaluated ontologies and mappings (**I**, **II** and **III**). For the Conference dataset we have selected only 5 pair of ontologies for which the reference mappings lead to more than five conservativity principle violations. Note that we count equivalence mappings as two subsumption mappings, and hence $\mathcal{M}$ represents subsumption mappings. Table 4 also shows the conservativity principle violations for the reference mappings (**IV** and **V**). For LargeBio and Library the number is expecially large using both our variant and the general notion of the conservativity principle.[13]

Tables 5 and 6 show the obtained results for our method using both the basic and optimised algorithms to add disjointness axioms.[14]

[13] In the SNOMED-NCI case no OWL 2 reasoner could succeed in classifying the integrated ontology via mappings [16], so we used the OWL 2 EL reasoner ELK [23] for providing a lower bound on the number of conservativity principle violations.

[14] The computation times of Steps 1-4 in Algorithm 1 were negligible with respect to the repair and disjointness addition times (t_r and t_d) and thus they were not included in the result tables.

Table 6. Results of our optimised method to detect and solve conservativity principle violations.

Dataset	$\mathcal{O}_1 \sim \mathcal{O}_2$	Solution size		Times		Remaining Violations	
		XII	**XIII**	**XIV**	**XV**	**XVI**	**XVII**
		#disj	$\lvert\mathcal{R}^{\approx}\rvert$	**t**$_d$**(s)**	**t**$_r$**(s)**	**consViol**	**diff**$^{\approx}$
Optique	**NPD∼BootsOnto**	214	41	2.54	0.17	0	0
LargeBio	**SNOMED∼NCI**	525,515	15,957	275	3,755	>411	>1,624
	FMA∼SNOMED	125,232	8,342	30	251	0	131
	FMA∼NCI	19,740	2,175	34	6.18	103	112
Anatomy	**MO∼NCI**$_{\textbf{Anat}}$	1,321	491	1.39	0.53	0	3
Library	**STW∼TheSoz**	42,045	3,058	4.93	41	0	40
Conference	**cmt∼confof**	11	6	0.05	0.01	0	0
	conference∼edas	8	6	0.07	0.01	0	0
	conference∼iasted	9	1	0.22	0.01	0	0
	confof∼ekaw	6	5	0.04	0.01	0	0
	edas∼iasted	7	4	0.21	0.02	1	1

We have run the experiments on a desktop computer with an *AMD Fusion A6-3670K* CPU and allocating 12 GB of RAM. The obtained results are summarized as follows:

i The number of added disjointness rules $disj$ (**VI**), as expected, is very large in the basic algorithm and the required time prohibitive (**VIII**) when involving SNOMED (it did not finish for SNOMED-NCI). This is clearly solved in our optimised algorithm that considerably reduces the number of necessary disjoitness rules (**XII**) and it requires only 275 seconds to compute them in the SNOMED-NCI case (**XIV**).

ii The computed repairs $\mathcal{R}^{\approx}$ (**VII** and **XIII**) using both the basic and optimised algorithms are of comparable size. This suggests that the large number of added disjointness in the basic algorithm does not have a negative impact (in terms of aggressiveness) on the repair process.

iii Repair times t$_r$ (**IX** and **XV**) are small and they do not represent a bottleneck in spite of the large number of added disjointness rules.

iv The conservativity principle violations using both algorithms and considering our variant (**X** and **XVI**) are completely removed in the Optique, Anatomy and Library cases, and almost completely removed in the Conference and LargeBio datasets.

v The number of missed violations is only slightly higher when considering the general notion of the conservativity principle (**XI** and **XVII**), which suggests that our (approximate) variant is also suitable in practice. Furthermore, in several test cases these violations are also almost removed.

vi The computed repairs $\mathcal{R}^{\approx}$, using both algorithms (**VII** and **XIII**), are rather aggressive and they can remove from 16% (Anatomy) up to 48% (Optique) of the mappings. In the Optique's use case, however, we follow a *better safe than sorry* approach and we prefer to remove as many violations as possible, rather than preserving potentially conflicting mapping sets.

In summary, the results suggest that our method to repair conservativity principle violations is suitable for Optique, and it is feasible in practice, even when considering the largest datasets of the OAEI.

6 Related Work

The conservativity principle problem, although indirectly, has been actively studied in the literature. For example, the assumption of disjointness was originally introduced by Schlobach [38] to enhance the repair of ontologies that were underspecified in terms of disjointness axioms. In [30], a similar assumption is followed in the context of repairing ontology mappings, where the authors restricted the number of disjointness axioms by using learning techniques [45]. These techniques, however, typically require a manually created training set. In [12] the authors present an interactive system to guide the expert user in the manual enrichment of the ontologies with disjointness axioms. In this paper, as in [45, 30, 12], we have also focused on the addition of a small set of disjointness axioms, since adding all possible disjointness may be unfeasible for large ontologies. However, our method does not require manual intervention. Furthermore, to address the scalability problem when dealing with large ontologies and mapping sets, our method relies on the propositional projection of the input ontologies.

Ontology matching systems have also dealt with the conservativity principle in order to improve the precision (with respect to a reference mapping set) of the computed mappings. For example, systems such as *ASMOV* [15], *Lily* [47] and *YAM++* [34] have implemented different heuristics and patterns to avoid violations of the conservativity principle. Another relevant approach has been presented in [2], where a set of sanity checks and best practices are proposed for computing ontology mappings. In this paper we present an elegant way to detect and solve conservativity principle violations by reducing the problem to a consistency principle violation problem. Concretely, we have reused and adapted the infrastructure provided by *LogMap* [17, 20]. However, other mapping repair systems, such as *Alcomo* [29] or *AML* [37], could be considered. Note that, to the best of our knowledge, these mapping repair systems have only focused on solving violations of the consistency principle.

The work presented in [26, 14, 27] deserves a special attention since they propose an opposite approach with respect to ours. Authors consider the violations of the conservativity principle as false positives, based on the potential incompleteness of the input ontologies. Hence, the correction strategy does not aim at removing mappings but at inserting subsumption axioms to the input ontologies to enrich their concept hierarchies. Authors in [35] also suggest that removing mapping may not be the best solution in a mapping repair process, and fixing the input ontologies may be an alternative.

Currently, in the Optique use case, we consider that the input ontologies are not modifiable. The query formulation ontology is based on the NPD ontology, which includes knowledge already agreed by the community, while the bootstrapped ontology is *directly* linked to the information represented in the database. Nevertheless, future extensions in Optique may consider appropriate the extension of the input ontologies.

7 Conclusions and Future Work

In this paper we have presented an approximate and fully-automatic method to detect and correct conservativity principle violations in practice. We have characterised the conservativity principle problem, following the assumption of disjointness, as a consistency principle problem. We have also presented an elegant and scalable way to detect and repair violations in the Horn propositional case. Thus, our method is incomplete and it may fail to detect and repair all violations. However, the conducted evaluation suggests that our method produces competitive results in practice. In the close future we plan to consider extensions of the current projection to Horn propositional logic while keeping the nice scalability properties of the current method.

The implemented method follows a "better safe than sorry" approach, which we currently consider suitable for the Optique project since we do not want ontology-to-ontology mappings to lead to unexpected results for the OBDA queries, as motivated in Section 3. Hence, we currently delegate complex relationhips between ontology entities and the database to the (hand-crafted) schema-to-ontology mappings, which will also play an important role in Optique. Nevertheless we do not discard in the future to explore alternative methods to detect and repair conservative principle violations. In particular, we plan to study the potential application of approaches based on graph-theory, in order to extend the detection and repair of conservativity principle violations. Strongly connected compontents of a graph representation of the subsumption relation between named concepts (as defined in [29]), for instance, may be used to capture violations between pairs of concepts already involved in a subsumption relationship.

Additionally, we will also consider exploring the use of learning techniques for the addition of disjointness axioms [45], and to involve the domain experts in the assessment/addition of such disjointness [18, 12]. This manual assessment may also be used to consider violations as false positives, as proposed in [26, 14, 27], and suggest them as candidate extensions of the input ontologies.

We consider that the proposed method has also potential in scenarios others than Optique. For instance, the authors in [28] apply ontology matching in a multi-agent system scenario in order to allow the exchange and extension of ontology-based *action plans* among agents. In such a context, violations of the conservativity principle should be taken into account and highly critical tasks should not be performed if violations are detected. In [44], authors present an ontology-based data integration (OBDI) system, which integrates ontology mapping and query reformulation techniques. As in Optique, mappings violating the conservativity principle may compromise the quality of the query results in the proposed OBDI system.

Finally, we have short-term plans for deployment in the Optique industry partners Statoil and Siemens. The techniques described in this paper have already been integrated within the "ontology and mapping management module" (see [24] for details about the Optique architecture).

Acknowledgements

This work was supported by the EU FP7 IP project Optique (no. 318338), the MIUR project CINA (Compositionality, Interaction, Negotiation, Autonomicity for the future ICT society) and the EPSRC project Score!. We also thank the unvaluable help provided by Bernardo Cuenca and Ian Horrocks. Finally, we are also very grateful for the support of the Optique colleagues that facilitated our understanding of the domain, especially: Dag Hovland, Evgeny Kharlamov, Dmitry Zheleznyakov and Martin G. Skjæveland.

References

1. Agrawal, R., Borgida, A., Jagadish, H.V.: Efficient Management of Transitive Relationships in Large Data and Knowledge Bases. In: ACM SIGMOD Conf. on Manag. of Data (1989)
2. Beisswanger, E., Hahn, U., et al.: Towards valid and reusable reference alignmentsten basic quality checks for ontology alignments and their application to three different reference data sets. J. Biomed. Semant. 3(Suppl 1), S4 (2012)
3. Bodenreider, O.: The Unified Medical Language System (UMLS): integrating biomedical terminology. Nucleic Acids Research 32, 267–270 (2004)
4. Borgida, A., Serafini, L.: Distributed Description Logics: Assimilating Information from Peer Sources. J. Data Sem. 1, 153–184 (2003)
5. Christophides, V., Plexousakis, D., Scholl, M., Tourtounis, S.: On Labeling Schemes for the Semantic Web. In: Int'l World Wide Web Conf. (WWW). pp. 544–555 (2003)
6. Cuenca Grau, B., Horrocks, I., Motik, B., Parsia, B., Patel-Schneider, P.F., Sattler, U.: OWL 2: The next step for OWL. J. Web Sem. 6(4), 309–322 (2008)
7. Cuenca Grau, B., Horrocks, I., Kazakov, Y., Sattler, U.: Modular Reuse of Ontologies: Theory and Practice. J. Artif. Intell. Res. 31, 273–318 (2008)
8. David, J., Euzenat, J., Scharffe, F., Trojahn, C.: The Alignment API 4.0. J. Sem. Web 2(1), 3–10 (2011)
9. Dowling, W.F., Gallier, J.H.: Linear-Time Algorithms for Testing the Satisfiability of Propositional Horn Formulae. J. Log. Prog. 1(3), 267–284 (1984)
10. Euzenat, J.: Semantic Precision and Recall for Ontology Alignment Evaluation. In: Int'l Joint Conf. on Artif. Intell. (IJCAI). pp. 348–353 (2007)
11. Euzenat, J., Meilicke, C., Stuckenschmidt, H., Shvaiko, P., Trojahn, C.: Ontology Alignment Evaluation Initiative: Six Years of Experience. J. Data Sem. 15, 158–192 (2011)
12. Ferré, S., Rudolph, S.: Advocatus Diaboli - Exploratory Enrichment of Ontologies with Negative Constraints. In: Int'l Conf. on Knowl. Eng. (EKAW). pp. 42–56 (2012)
13. Grau, B.C., Dragisic, Z., Eckert, K., et al.: Results of the Ontology Alignment Evaluation Initiative 2013. In: Ontology Matching (OM) (2013)
14. Ivanova, V., Lambrix, P.: A Unified Approach for Aligning Taxonomies and Debugging Taxonomies and their Alignments. In: Eur. Sem. Web Conf. (ESWC), pp. 1–15. Springer (2013)
15. Jean-Mary, Y.R., Shironoshita, E.P., Kabuka, M.R.: Ontology Matching With Semantic Verification. J. Web Sem. 7(3), 235–251 (2009)
16. Jiménez-Ruiz, E., Cuenca Grau, B., Horrocks, I.: On the feasibility of using OWL 2 DL reasoners for ontology matching problems. In: OWL Reasoner Evaluation Workshop (2012)
17. Jiménez-Ruiz, E., Cuenca Grau, B.: LogMap: Logic-based and Scalable Ontology Matching. In: Int'l Sem. Web Conf. (ISWC). pp. 273–288 (2011)
18. Jiménez-Ruiz, E., Cuenca Grau, B., Horrocks, I., Berlanga, R.: Ontology integration using mappings: Towards getting the right logical consequences. In: Eur. Sem. Web Conf. (2009)
19. Jiménez-Ruiz, E., Cuenca Grau, B., Horrocks, I., Berlanga, R.: Logic-based Assessment of the Compatibility of UMLS Ontology Sources. J. Biomed. Semant. 2(Suppl 1), S2 (2011)
20. Jiménez-Ruiz, E., Cuenca Grau, B., Zhou, Y., Horrocks, I.: Large-scale Interactive Ontology Matching: Algorithms and Implementation. In: Eur. Conf. on Artif. Intell. (ECAI) (2012)
21. Jiménez-Ruiz, E., Meilicke, C., Grau, B.C., Horrocks, I.: Evaluating Mapping Repair Systems with Large Biomedical Ontologies. In: Description Logics. pp. 246–257 (2013)
22. Kalyanpur, A., Parsia, B., Horridge, M., Sirin, E.: Finding all justifications of OWL DL entailments. Int'l Sem. Web Conf. (ISWC) pp. 267–280 (2007)
23. Kazakov, Y., Krötzsch, M., Simancik, F.: Concurrent Classification of EL Ontologies. In: Int'l Sem. Web Conf. (ISWC). pp. 305–320 (2011)

24. Kharlamov, E., Jiménez-Ruiz, E., Zheleznyakov, D., et al.: Optique: Towards OBDA Systems for Industry. In: Eur. Sem. Web Conf. (ESWC) Satellite Events. pp. 125–140 (2013)
25. Konev, B., Walther, D., Wolter, F.: The Logical Difference Problem for Description Logic Terminologies. In: Int'l Joint Conf. on Automated Reasoning (IJCAR). pp. 259–274 (2008)
26. Lambrix, P., Dragisic, Z., Ivanova, V.: Get My Pizza Right: Repairing Missing Is-a Relations in $\mathcal{ALC}$ Ontologies. In: Semantic Technology, pp. 17–32. Springer (2013)
27. Lambrix, P., Liu, Q.: Debugging the Missing Is-a Structure Within Taxonomies Networked by Partial Reference Alignments. Data Knowl. Eng. (DKE) 86, 179–205 (2013)
28. Mascardi, V., Ancona, D., Barbieri, M., Bordini, R.H., Ricci, A.: CooL-AgentSpeak: Endowing AgentSpeak-DL Agents with Plan Exchange and Ontology Services. Web Intelligence and Agent Systems 12(1), 83–107 (2014)
29. Meilicke, C.: Alignments Incoherency in Ontology Matching. Ph.D. thesis, University of Mannheim (2011)
30. Meilicke, C., Völker, J., Stuckenschmidt, H.: Learning Disjointness for Debugging Mappings between Lightweight Ontologies. In: Int'l Conf. on Knowl. Eng. (EKAW). pp. 93–108 (2008)
31. Melnik, S., Garcia-Molina, H., Rahm, E.: Similarity Flooding: A Versatile Graph Matching Algorithm and Its Application to Schema Matching. In: IEEE Int'l Conf. on Data Eng. (2002)
32. Motik, B., Shearer, R., Horrocks, I.: Hypertableau Reasoning for Description Logics. J. Artif. Intell. Res. (JAIR) 36, 165–228 (2009)
33. Nebot, V., Berlanga, R.: Efficient Retrieval of Ontology Fragments Using an Interval Labeling Scheme. Inf. Sci. 179(24), 4151–4173 (2009)
34. Ngo, D., Bellahsene, Z.: YAM++ : A Multi-strategy Based Approach for Ontology Matching Task. In: Int'l Conf. on Knowl. Eng. (EKAW). pp. 421–425 (2012)
35. Pesquita, C., Faria, D., Santos, E., Couto, F.M.: To repair or not to repair: reconciling correctness and coherence in ontology reference alignments. In: Ontology Matching (OM) (2013)
36. Reiter, R.: A Theory of Diagnosis from First Principles. Artif. Intell. 32(1) (1987)
37. Santos, E., Faria, D., Pesquita, C., Couto, F.: Ontology Alignment Repair Through Modularization and Confidence-based Heuristics. arXiv:1307.5322 preprint (2013)
38. Schlobach, S.: Debugging and Semantic Clarification by Pinpointing. In: Eur. Sem. Web Conf. (ESWC), pp. 226–240. Springer (2005)
39. Schlobach, S., Cornet, R.: Non-standard Reasoning Services for the Debugging of Description Logic Terminologies. In: Int'l Joint Conf. on Artif. Intell. (IJCAI). pp. 355–362 (2003)
40. Shvaiko, P., Euzenat, J.: Ontology Matching: State of the Art and Future Challenges. IEEE Transactions on Knowl. and Data Eng. (TKDE) (2012)
41. Skjæveland, M.G., Lian, E.H., Horrocks, I.: Publishing the Norwegian Petroleum Directorate's FactPages as Semantic Web Data. In: Int'l Sem. Web Conf. (ISWC) (2013)
42. Soylu, A., et al.: A Preliminary Approach on Ontology-Based Visual Query Formulation for Big Data. In: 7th Research Conf. on Metadata and Semantics Research (MTSR) (2013)
43. Suntisrivaraporn, B., Qi, G., Ji, Q., Haase, P.: A Modularization-Based Approach to Finding All Justifications for OWL DL Entailments. In: Asian Sem. Web Conf. (ASWC) (2008)
44. Tian, A., Sequeda, J., Miranker, D.P.: QODI: Query as Context in Automatic Data Integration. In: Int'l Sem. Web Conf. (ISWC). pp. 624–639 (2013)
45. Völker, J., Vrandecic, D., Sure, Y., Hotho, A.: Learning Disjointness. In: Eur. Sem. Web Conf. (ESWC). pp. 175–189 (2007)
46. Šváb, O., Svátek, V., Berka, P., Rak, D., Tomášek, P.: OntoFarm: Towards an Experimental Collection of Parallel Ontologies. In: Int'l Sem. Web Conf. (ISWC). Poster Session (2005)
47. Wang, P., Xu, B.: Debugging Ontology Mappings: A Static Approach. Computing and Informatics 27(1), 21–36 (2012)
48. Zhang, S., Mork, P., Bodenreider, O.: Lessons Learned from Aligning two Representations of Anatomy. In: Int'l Conf. on Principles of Knowl. Repr. and Reasoning (KR) (2004)

Appendix G

ISWC 2014: Repair in Ontology Alignments

This appendix reports the paper:

– Daniel Faria, Ernesto Jimenez-Ruiz, Catia Pesquita, Emanuel Santos, and Francisco M. Couto. Towards annotating potential incoherences in BioPortal mappings. In Proceedings of the International Semantic Web Conference 2014

Towards annotating potential incoherences in BioPortal mappings

Daniel Faria[1], Ernesto Jiménez-Ruiz[2], Catia Pesquita[1,3], Emanuel Santos[3], and Francisco M. Couto[1,3]

[1] LASIGE, Faculdade de Ciências, Universidade de Lisboa, Portugal
[2] Department of Computer Science, University of Oxford, UK
[3] Departamento de Informática, Faculdade de Ciências, Universidade de Lisboa, Portugal

Abstract. BioPortal is a repository for biomedical ontologies that also includes mappings between them from various sources. Considered as a whole, these mappings may cause logical errors, due to incompatibilities between the ontologies or even erroneous mappings.
We have performed an automatic evaluation of BioPortal mappings between 19 ontology pairs using the mapping repair systems of LogMap and AgreementMakerLight. We found logical errors in 11 of these pairs, which on average involved 22% of the mappings between each pair. Furthermore, we conducted a manual evaluation of the repair results to identify the actual sources of error, verifying that erroneous mappings were behind over 60% of the repairs.
Given the results of our analysis, we believe that annotating BioPortal mappings with information about their logical conflicts with other mappings would improve their usability for semantic web applications and facilitate the identification of erroneous mappings. In future work, we aim to collaborate with BioPortal developers in extending BioPortal with these annotations.

1 Motivation

OWL ontologies are extensively used in biomedical information systems. Prominent examples of biomedical ontologies are the Gene Ontology [1], the National Cancer Institute Thesaurus (NCIT) [14] and the Foundational Model of Anatomy (FMA) [32].

Despite some community efforts to ensure a coordinated development of biomedical ontologies [38], many ontologies are being developed independently by different groups of experts and, as a result, they often cover the same or related subjects, but follow different modeling principles and use different entity naming schemes. Thus, to integrate data among applications, it is crucial to establish correspondences (called mappings) between the entities of the ontologies they use.

In the last ten years, the semantic web and bioinformatics research communities have extensively investigated the problem of (semi-)automatically computing correspondences between independently developed ontologies, which is usually referred to as the *ontology matching problem*. Resulting from this effort are the growing number of ontology matching systems in development [8,7,37] and the large mapping repositories that have been created (e.g., [2,10]).

One such repository, BioPortal [10,33], is a coordinated community effort which currently provides access to more than 370 biomedical ontologies and over 12 million mappings between them.[4] While not all BioPortal ontologies were originally OWL ontologies (e.g., some were developed in OBO format[5]), many have been (or can be) converted to OWL [15]. Mappings in BioPortal are either generated automatically by a sophisticated lexical matcher [13] or added manually by domain experts through the Web interface or the REST APIs [29].

OWL ontologies have well-defined semantics [4] and thus the integration of independently developed ontologies via a set of mappings (i.e., an alignment) may lead to logical errors such as unsatisfiablities [25]. BioPortal, however, explicitly supports the idea that alternative (i.e., created for different purposes) mapping sets may co-exist and that they could potentially contradict each other [29].

While it is true that many logical errors in alignments are caused by incompatibilities between the ontologies they map [19,31], some may be caused by erroneous mappings. Furthermore, logical soundness may be critical to some semantic web applications that integrate two or more ontologies [31]. For these reasons, we consider that it would be advantageous to enrich BioPortal mappings with *annotations* about potential logical conflicts with other mappings. This would improve the usability of BioPortal mappings for semantic web applications and domain users, and facilitate the identification of erroneous mappings and potential errors in the ontologies themselves.

In this paper we quantify the logical errors in the BioPortal mappings among several ontologies by applying mapping repair algorithms. Furthermore, we manually analyze a subset of the identified conflicting mappings in order to qualify the causes of the errors. Our goal is to show the importance of identifying (and annotating) logical conflicts in BioPortal and the role ontology mapping repair algorithms may play in that task.

The rest of the paper is organized as follows: Section 2 describes how mappings are represented in BioPortal; Section 3 introduces the concept of mapping repair and presents the repair algorithms used in this study; Section 4 details the automatic and manual evaluations we conducted and presents and discusses their results; and finally, Section 5 presents some conclusions and future work lines.

2 Mappings in BioPortal

Mappings are treated as first-class citizens in BioPortal [29,12], as it enables the querying, upload, and retrieval of all mappings between all of its ontologies. A survey conducted in 2009 revealed that 33% of BioPortal ontologies had 50% of their entities mapped to other ontologies [12], which indicates that BioPortal ontologies are highly interconnected.

The number of mappings in BioPortal has grown quickly in recent years, from 30,000 mappings between 20 ontologies in 2008 [29] to 9 million mappings between 302 ontologies in 2012 [33]. At the time of writing this paper, there were approximately 13 million mappings between 373 ontologies.

[4] BioPortal: `https://bioportal.bioontology.org/`

[5] `http://www.geneontology.org/GO.format.obo-1_4.shtml`

Mappings in BioPortal are represented as a 4-tuple of the form $\langle e_1, e_2, Rel, Ann\rangle$, where e_1, e_2 are the URIs of two entities from the vocabulary of two BioPortal ontologies, Rel is the semantic relationship between them, and Ann is a set of annotations or metadata associated to the mapping. The relation Rel can be of one of the following types:[6] *skos:exactMatch*, *skos:relatedMatch*, *skos:closeMatch*, *skos:narrowMatch*, *skos:broadMatch*. Ann includes, among other details, important provenance information about the mapping such as: origin (e.g., user-defined or alignment system employed), application context, creator, and creation date.

According to BioPortal authors [33], the statistics about mapping origin are the following: *(i)* 64.96% of the mappings were created by the lexical matcher LOOM [13]; *(ii)* 32.48% of the mappings had UMLS [2] as origin; *(iii)* 2.41% represented mappings between entities with the same URI; *(iv)* 0.02% came from *Xref* OBO Mappings; *(v)* finally 0.13% of the mappings were submitted by users.

Mappings between entities with the same URI are labeled *skos:exactMatch* by BioPortal, LOOM and UMLS mappings are labeled *skos:closeMatch*, and *Xref* OBO Mappings are labeled *skos:relatedMatch*. User submitted mappings can be labeled with any of the relation types listed above.

BioPortal mappings can be retrieved via its REST API, being straightforward to identify the entities involved in the mapping, its origin, and the source ontologies.[7]

In this paper, we have focused only on *skos:closeMatch* mappings, which account for the large majority of BioPortal mappings. We represented these mappings as OWL 2 equivalence axioms since that is typically the semantic relation they convey (the tag *skos:closeMatch* is used to link concepts that can be used interchangeably, at least in a given context). Mapping annotations Ann have (optionally) been represented as OWL 2 axiom annotations. This representation of mappings enables the reuse of the extensive range of OWL 2 reasoning infrastructure that is currently available. Note that alternative formal semantics for ontology mappings have been proposed in the literature (e.g., [3,6,28]).

3 Mapping repair

The ontology resulting from the integration of $\mathcal{O}_1$ and $\mathcal{O}_2$ via a set of mappings $\mathcal{M}$, may entail axioms that do not follow from $\mathcal{O}_1$, $\mathcal{O}_2$, or $\mathcal{M}$ alone. These new semantic consequences can be captured using the notion of *deductive difference* [24,18], and can be divided into desired and undesired entailments. Undesired entailments are typically introduced due to erroneous mappings in $\mathcal{M}$. However, even if all mappings in $\mathcal{M}$ are correct, undesired entailments may occur due to conflicting descriptions between the overlapping entities in $\mathcal{O}_1$ and $\mathcal{O}_2$. Undesired entailment can be divided into two groups: entailments causing unsatisfiable classes, which can be easily detected using (automatic) logical reasoning; and entailments not causing unsatisfiable classes, which require domain knowledge to decide whether they are indeed undesired. In this paper we only focus on the first group of undesired entailments.

[6] `http://www.bioontology.org/wiki/index.php/BioPortal_Mappings`
[7] `http://data.bioontology.org/documentation#Mapping`

A set of mappings that leads to unsatisfiable classes in $\mathcal{O}_1 \cup \mathcal{O}_2 \cup \mathcal{M}$ is referred to as *incoherent* w.r.t. $\mathcal{O}_1$ and $\mathcal{O}_2$ [26].

Definition 1 (Mapping Incoherence). *A set of mappings $\mathcal{M}$ is incoherent with respect to $\mathcal{O}_1$ and $\mathcal{O}_2$, if there exists a class A in the signature of $\mathcal{O}_1 \cup \mathcal{O}_2$ such that $\mathcal{O}_1 \cup \mathcal{O}_2 \not\models A \sqsubseteq \bot$ and $\mathcal{O}_1 \cup \mathcal{O}_2 \cup \mathcal{M} \models A \sqsubseteq \bot$.*

An incoherent set of mappings $\mathcal{M}$ can be fixed by removing mappings from $\mathcal{M}$. This process is referred to as *mapping repair* (or repair for short).

Definition 2 (Mapping Repair). *Let $\mathcal{M}$ be an incoherent set of mappings w.r.t. $\mathcal{O}_1$ and $\mathcal{O}_2$. A set of mappings $\mathcal{R} \subseteq \mathcal{M}$ is a mapping repair for $\mathcal{M}$ w.r.t. $\mathcal{O}_1$ and $\mathcal{O}_2$ if $\mathcal{M} \setminus \mathcal{R}$ is coherent w.r.t. $\mathcal{O}_1$ and $\mathcal{O}_2$.*

A trivial repair is $\mathcal{R} = \mathcal{M}$, since an empty set of mappings is obviously coherent. Nevertheless, the objective is to remove as few mappings as possible. Minimal (mapping) repairs are typically referred to in the literature as *mapping diagnosis* [25].

In the literature there are different approaches to compute a repair or diagnosis for an incoherent set of mappings. Early approaches were based on Distributed Description Logics (DDL) (e.g. [27,28,30]). The work presented in [30] deserves special mention, as it reports on a preliminary coherence evaluation of BioPortal mappings using DDL.[8] The authors, however, emphasized the problems of efficiency of the coherence checking task due to the reasoning complexity of DDL and suggest the use of approximate techniques in the future.

Alternatively, if mappings are represented as OWL 2 axioms, mapping repairs can also be computed using the state-of-the-art approaches for debugging and repairing inconsistencies in OWL 2 ontologies, which rely on the extraction of justifications for the unsatisfiable classes (e.g. [36,22,39,18]). However, justification-based technologies do not scale when the number of unsatisfiabilities is large (a typical scenario in mapping repair problems [16]).

To address this scalability issue, mapping repair systems usually compute an *approximate repair* using incomplete reasoning techniques (e.g. [17,25,9]). An approximate repair $\mathcal{R}^{\approx}$ does not guarantee that $\mathcal{M} \setminus \mathcal{R}^{\approx}$ is coherent, but it will (in general) reduce significantly the number of unsatisfiabilities caused by the original mappings $\mathcal{M}$. Indeed, approximate repair techniques have been successfully applied to audit the UMLS metathesaurus [19,17].

In this paper, we have applied the *approximate* mapping repair techniques implemented in LogMap [17,20,21] and AgreementMakerLight (AML) [9,35] to the BioPortal mappings. As described in Section 2, we have represented the BioPortal mappings as OWL 2 equivalence axioms. Note that, although both LogMap and AML were originally implemented as ontology matching systems, they can also operate as a standalone mapping repair systems. From this point onwards, we will refer to LogMap's and AML's repair modules as LogMap-Repair and AML-Repair respectively.

Algorithm 1 AML-Repair algorithm

```
Input: $\mathcal{O}_1$, $\mathcal{O}_2$: input ontologies; $\mathcal{M}$: input mappings
Output: $\mathcal{M}'$: output mappings; $\mathcal{R}^{\approx}$: approximate mapping repair; $\mathcal{CS}$: identified conflicting sets; $\mathcal{M}_{\mathcal{CS}}$: mappings involved in conflicting sets
 1: $\mathcal{M}' := \mathcal{M}$
 2: $\mathcal{R}^{\approx} := \emptyset$
 3: $\langle \mathcal{O}'_1, \mathcal{O}'_2$, Checkset$\rangle$ := $\mathsf{BuiltCoreFragments}(\mathcal{O}_1, \mathcal{O}_2, \mathcal{M}')$
 4: $\mathcal{CS} := \mathsf{ConflictSets}(\mathcal{O}'_1, \mathcal{O}'_2, M'$, Checkset)
 5: $\mathcal{M}_{\mathcal{CS}} := \mathsf{MappingsInConflictSets}(\mathcal{CS})$
 6: $\mathcal{CS}' := \mathcal{CS}$
 7: while $|\mathcal{CS}'| > 0$ do
 8:     $w := \mathsf{SelectMappingToRemove}(\mathcal{CS}')$
 9:     $\mathcal{CS}' := \mathsf{RemoveMapping}(\mathcal{CS}', w)$
10:     $\mathcal{M}' := \mathcal{M}' \setminus \{w\}$;
11:     $\mathcal{R}^{\approx} := \mathcal{R}^{\approx} \cup \{w\}$
12: end while
13: return $\langle \mathcal{M}', \mathcal{R}^{\approx}, \mathcal{CS}, \mathcal{M}_{\mathcal{CS}} \rangle$
```

3.1 Mapping repair using AML-Repair

The pseudocode of the algorithm implemented by AML-Repair is described in Algorithm 1. The algorithm is divided in three main tasks:

1. The computation of the core fragments (see [34]) (step 3);
2. The search for all (minimal) conflicting sets of mappings $\mathcal{CS}$, i.e. mappings that lead to an incoherence (step 4);
3. The resolution of incoherences using a heuristic to minimize the set of mappings removed from every conflicting set (step 8 to 11);
4. The algorithm outputs a set of repaired mappings $\mathcal{M}'$, an approximate mapping repair $\mathcal{R}^{\approx}$, conflicting sets of mappings $\mathcal{CS}$, and the set of all mappings involved in at least one conflicting set $\mathcal{M}_{\mathcal{CS}}$.

AML-Repair implementation is based on a modularization of the input ontologies, called core fragments, that only contains the necessary classes and relations to detect all existing incoherences [34]. This modularization is computed by the BuildCoreFragments method (Step 3 of Algorithm 1), which also computes a minimal set of classes (the Checkset) that need to be checked for incoherences.

AML-Repair determines subsumption relations between atomic classes syntactically (i.e., without using an OWL 2 Reasoner) and it also considers disjointness axioms between atomic classes. Unlike LogMap-Repair, equivalence mappings are considered indivisible units and are never split into two subsumption mappings. Thus, an input mapping is either removed or kept in the alignment during the repair procedure.

The ConflictSets method (step 4) returns all mapping sets that will lead to an incoherence by doing a full depth-first search in the core fragments structure for each class in the Checkset. This way, AML-Repair determines all minimal sets of mappings, called

[8] To the best of our knowledge, no automatic repair was conducted.

Algorithm 2 LogMap-Repair algorithm based on Horn propositional reasoning

Input: $\mathcal{O}_1$, $\mathcal{O}_2$: input ontologies; $\mathcal{M}$: input mappings
Output: $\mathcal{M}'$: output mappings; $\mathcal{R}^{\approx}$: approximate mapping repair; $\mathcal{CG}$: conflicting groups; $\overline{\mathcal{M}_{\mathcal{CG}}}$: mapping average in conflicting groups

```
 1: $\mathcal{M}' := \mathcal{M}$
 2: $\mathcal{R}^{\approx} := \emptyset$
 3: $\mathcal{CG} := \emptyset$
 4: $\langle \mathcal{P}_1, \mathcal{P}_2 \rangle := \mathsf{PropEncoding}(\mathcal{O}_1, \mathcal{O}_2)$
 5: for each $C \in \mathsf{OrderedVariables}(\mathcal{P}_1 \cup \mathcal{P}_2)$ do
 6:     $\mathcal{P}_C := \mathcal{P}_1 \cup \mathcal{P}_2 \cup \mathcal{M}' \cup \{\mathsf{true} \rightarrow C\}$
 7:     $\langle sat, \mathcal{M}_{\perp} \rangle := \mathsf{DowlingGallier}(\mathcal{P}_C)$
 8:     if $sat = \mathsf{false}$ then
 9:         $\mathcal{CG} := \mathcal{CG} \cup \{\mathcal{M}_{\perp}\}$
10:         $Rep := \emptyset$
11:         $rep\_size := 1$
12:         repeat
13:             for each subset $\mathcal{R}_C$ of $\mathcal{M}_{\perp}$ of size $rep\_size$ do
14:                 $sat := \mathsf{DowlingGallier}(\mathcal{P}_C \setminus \mathcal{R}_C)$
15:                 if $sat = \mathsf{true}$ then $Rep := Rep \cup \{\mathcal{R}_C\}$
16:             end for
17:             $rep\_size := rep\_size + 1$
18:         until $Rep \neq \emptyset$
19:         $\mathcal{R}_C$ := element of $Rep$ with minimum aggregated confidence.
20:         $\mathcal{M}' := \mathcal{M}' \setminus \mathcal{R}_C$
21:         $\mathcal{R}^{\approx} := \mathcal{R}^{\approx} \cup \mathcal{R}_C$
22:     end if
23: end for
24: $\overline{\mathcal{M}_{\mathcal{CG}}} := \mathsf{AverageMappingsInConflictGroups}(\mathcal{CG})$
25: return $\langle \mathcal{M}', \mathcal{R}^{\approx}, \mathcal{CG}, \overline{\mathcal{M}_{\mathcal{CG}}} \rangle$
```

conflicting sets $\mathcal{CS}$, which cause the incoherences. Since conflicting sets are minimal, a conflicting set is resolved if at least one of its mappings is removed. The algorithm also keeps the set $\mathcal{M}_{\mathcal{CS}}$ containing all mappings involved in a conflicting set (Step 5).

AML-Repair aims to minimize the number of removed mappings by determining a minimal set of mappings that intersect all conflict sets. Given that computing this set is NP-Complete, AML-Repair uses an efficient heuristic procedure that consists of iteratively removing the mappings that belong to the highest number of conflicting sets (as identified in Step 8 of Algorithm 1), and in case of tie, those that have the lowest confidence values. This strategy typically produces near-optimal results.

3.2 Mapping repair using LogMap-Repair

Algorithm 2 shows the pseudocode of the algorithm implemented by LogMap-Repair. Steps 1-3 initialise the output variables. LogMap-Repair encodes the input ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$ as Horn propositional theories $\mathcal{P}_1$ and $\mathcal{P}_2$ (Step 4) and exploits this encoding to subsequently detect unsatisfiable classes in an efficient and sound way during the repair process. The theory $\mathcal{P}_1$ (resp. $\mathcal{P}_2$) consists of the following Horn rules:

– A rule $A \rightarrow B$ for all distinct classes A, B such that A is subsumed by B in $\mathcal{O}_1$ (resp. in O_2); subsumption relations can be determined using either an OWL 2 reasoner, or syntactically (in an incomplete way).
– Rules $A_i \wedge A_j \rightarrow \mathsf{false}$ $(1 \leq i < j \leq n)$ for each disjointness axiom of the form $DisjointClasses(A_1, \ldots, A_n)$.
– A rule $A_1 \wedge \ldots \wedge A_n \rightarrow B$ for each subclass or equivalence axiom having the intersection of $A_1, \ldots A_n$ as subclass expression and B as superclass.

In Step 5, propositional variables in $\mathcal{P}_1$ (resp. in $\mathcal{P}_2$) are ordered such that a variable C in $\mathcal{P}_1$ (resp. in $\mathcal{P}_2$) comes before D whenever D is subsumed by C in $\mathcal{O}_1$ (resp. in $\mathcal{O}_2$). This is a well-known repair strategy: subclasses of an unsatisfiable class are also unsatisfiable and hence before repairing an unsatisfiable class one first needs to repair its superclasses. Satisfiability of a propositional variable C is determined by checking satisfiability of the propositional theory $\mathcal{P}_C$ (Step 6) consisting of *(i)* the rule $(\mathtt{true} \rightarrow C)$; *(ii)* the propositional representations $\mathcal{P}_1$ and $\mathcal{P}_2$; and *(iii)* the current set of output mappings $\mathcal{M}'$ (seen as propositional implications). Note that LogMap-Repair splits equivalence mappings into two equivalent subsumption mappings.

LogMap-Repair implements the classical Dowling-Gallier algorithm for propositional Horn satisfiability [5,11]. LogMap-Repair's implementation of Dowling-Gallier's algorithm also records all mappings potentially involved in an unsatisfiability. Thus, a call to Dowling-Gallier returns a satisfiability value sat and, optionally, the (overestimated) group of conflicting mappings $\mathcal{M}_\perp$ (see Steps 7 and 14). For statistical purposes, the set $\mathcal{CG}$ keeps all conflicting groups for the identified unsatisfiable classes (Step 9). An unsatisfiable class C is repaired by discarding conflicting mappings for C (Steps 10 to 21). Thus, subsets $\mathcal{R}_C$ of $\mathcal{M}_\perp$ of increasing size are then identified until a repair is found (Steps 12-18). Note that, LogMap-Repair does not compute a diagnosis for the unsatisfiable class C but rather the repairs of smallest size. If several repairs of a given size exist, the one with the lowest aggregated confidence is selected according to the confidence values assigned to mappings (Step 19). Steps 20 and 21 update the output mappings $\mathcal{M}'$ and the approximate mapping repair $\mathcal{R}^{\approx}$ by extracting and adding $\mathcal{R}_C$, respectively. Finally, Step 24 calculates the average number of mappings in each identified conflicting group $\mathcal{CG}$.

Algorithm 2 ensures that $\mathcal{P}_1 \cup \mathcal{P}_2 \cup \mathcal{M}' \cup \{\mathtt{true} \rightarrow C\}$ is satisfiable for each C occurring in $\mathcal{P}_1 \cup \mathcal{P}_2$. The propositional encoding of $\mathcal{O}_1$ and $\mathcal{O}_2$ is, however, incomplete and hence the algorithm does not ensure satisfiability of each class in $\mathcal{O}_1 \cup \mathcal{O}_2 \cup \mathcal{M}'$. Nevertheless, the number of unsatisfiable classes remaining after computing an approximate repair $\mathcal{R}^{\approx}$ is typically small.

4 Evaluation

In order to evaluate the coherence of BioPortal mappings, we manually selected 19 ontology pairs from BioPortal such that *(i)* each pair had at least 500 mappings listed in BioPortal, *(ii)* at least one of the ontologies in the pair contained disjointness clauses between their classes, and *(iii)* the domain of both ontologies was biomedical. The purpose of the first two criteria is to exclude ontology pairs that are uninteresting from an (automatic) mapping repair perspective, whereas the third criterion ensures that we are

Table 1: Ontologies comprising the 19 ontology pairs selected.

Ontology	Acronym	# Classes	Source
Bone Dysplasia Ontology	BDO	13,817	BioPortal
Cell Culture Ontology	CCONT	14,663	BioPortal
Experimental Factor Ontology	EFO	14,499	BioPortal
Human Developmental Anatomy Ontology, timed ver.	EHDA	8,340	OBO Foundry
Cardiac Electrophysiology Ontology	EP	81,957	BioPortal
Foundational Model of Anatomy	FMA	83,280	BioPortal
Mouse Adult Gross Anatomy Ontology	MA	3,205	OBO Foundry
NCI Thesaurus	NCIT	105,347	BioPortal
Online Mendelian Inheritance in Man	OMIM	76,721	BioPortal
Sleep Domain Ontology	SDO	1,382	BioPortal
SNP ontology	SNP	2,206	BioPortal
Sequence Types and Features Ontology	SO	2,021	BioPortal
Teleost Anatomy Ontology	TAO	3,372	OBO Foundry
Uber Anatomy Ontology	UBERON	15,773	OBO Foundry
Zebrafish Anatomy and Development Ontology	ZFA	2,955	OBO Foundry

Table 2: BioPortal mappings for the selected ontology pairs

Ontology Pair	Listed Mappings	Retrieved Mappings	Actual Mappings	Unsat. Classes
BDO-NCIT	1,637	1,636	1,636	34,341
CCONT-NCIT	2,815	2,813	2,097 (-19)	50,304
EFO-NCIT	3,289	3,287	2,507	60,347
EHDA-FMA	3,731	2,496	2,496	0
EP-FMA	79,497	78,489	78,489	210
EP-NCIT	2,468	2,465	2,465 (-1)	14,687 (-1)
MA-FMA	5,491	961	961	850
OMIM-NCIT	5,198	5,198	5,178	70,172
SDO-EP	662	135	135	44
SDO-FMA	593	529	529 (-1)	0
SNPO-SO	2,168	2,150	2,028 (-1)	0
UBERON-FMA	2,233	1,932	1,932	4,753
ZFA-CCONT	532	437	333	0
ZFA-EFO	773	538	427	913
ZFA-EHDA	2,595	1,809	1,809	0
ZFA-FMA	1,240	265	265	0
ZFA-MA	1,639	129	129	0
ZFA-TAO	1,737	1,524	1,521	0
ZFA-UBERON	817	724	724	104

able to manually evaluate the repair results as they lie within our domain of expertise. This selection was not exhaustive, as our goal was merely to select a substantial and representative set of ontology pairs.

The 15 ontologies comprising these 19 pairs are listed in Table 1. We retrieved the latest OWL version of each ontology from BioPortal, except for the ontologies that

Algorithm 3 Automatic repair evaluation of BioPortal mappings

```
Input: $\mathcal{O}_1$, $\mathcal{O}_2$: two BioPortal ontologies; $\mathcal{M}$: the set of BioPortal mappings between them
1: Compute all conflict sets of mappings $\mathcal{CS}$, the total number of mappings involved in conflicts
   $\mathcal{M}_{\mathcal{CS}}$, and the approximate repair $\mathcal{R}^{\approx}$ using AML-Repair system    ▷ See Algorithm 1
2: Get unsatisfiable classes of $\mathcal{O}_1 \cup \mathcal{O}_2 \cup \mathcal{M} \setminus \mathcal{R}^{\approx}$ using ELK reasoner
3: Compute the conflicting mapping groups $\mathcal{CG}$ per unsatisfiability, the average number of map-
   pings per conflict group $\overline{\mathcal{M}_{\mathcal{CG}}}$, and the approximate repair $\mathcal{R}^{\approx}$ using LogMap-Repair system
   ▷ See Algorithm 2
4: Get unsatisfiable classes of $\mathcal{O}_1 \cup \mathcal{O}_2 \cup \mathcal{M} \setminus \mathcal{R}^{\approx}$ using ELK reasoner
```

were only available in OBO format. Because AML is currently not set-up to handle ontologies in the OBO format, we retrieved the latter from the OBO Foundry[9] [38] in OWL format (making sure the versions matched those in BioPortal).

We implemented a script that, given a pair of ontologies, uses BioPortal's REST API to retrieve all mappings between those ontologies. We focused only on *skos:closeMatch* mappings and we represented them as OWL 2 equivalence axioms. We did not consider *skos:exactMatch* mappings since they represent correspondences between entities with the same URI, which in OWL ontologies are considered equivalent (even though the equivalence between them is not explicitly defined). We also excluded a few mappings that had only a *null* source or involved only one entity.

The mappings between the 19 selected ontology pairs are listed in Table 2. We verified that the number of retrieved mappings did not match the number of mappings listed in the BioPortal website, and that sometimes the discrepancy was large (i.e., not accounted for by the small fraction of mappings we excluded). BioPortal developers confirmed that there is indeed an inconsistency between the metrics and the available mappings. Furthermore, in several cases, some of the mappings retrieved pointed to classes that were not found in the ontologies (possibly obsolete classes), so the actual mappings between the ontologies were less than those retrieved. Additionally, in some cases less mappings were found when using the Jena API to read the ontologies (used by AML) than when using the OWL API (used by LogMap). The difference between the two is shown in parenthesis in Table 2.

Finally, we computed the satisfiability of each alignment with the OWL 2 EL reasoner ELK [23], finding several unsatisfiable classes in 11 of the alignments. We opted for ELK for the sake of efficiency, given the size of some of the ontologies. ELK is incomplete and thus the identified unsatisfiabilities represent a lower bound of the actual number of such logical errors.

4.1 Automatic Repair Evaluation

For each of the 11 ontology pairs that had incoherent mapping sets (as detected by ELK and listed in Table 2), we conducted the evaluation detailed in Algorithm 3. The results we obtained are shown in Table 3. Note that LogMap-Repair splits equivalence mappings into two subsumption mappings, so the value of $\mathcal{R}^{\approx}$ is not directly comparable with AML-Repair (the latter should be doubled to compare it with the former).

[9] `http://www.obofoundry.org/`

Table 3: Automatic mapping repair using AML-Repair and LogMap-Repair

Ontology Pairs	$\mathcal{M}$	**AML-Repair**				**LogMap-Repair**			
		$\lvert\mathcal{CS}\rvert$	$\lvert\mathcal{M}_{\mathcal{CS}}\rvert$	$\lvert\mathcal{R}^{\approx}\rvert$	**Unsat.**	$\lvert\mathcal{CG}\rvert$	$\overline{\mathcal{M}_{\mathcal{CG}}}$	$\lvert\mathcal{R}^{\approx}\rvert$	**Unsat.**
BDO-NCIT	1,636	1,649	1,374 (84%)	53	0	125	3.2	154	0
CCONT-NCIT	2,097	1,197	1,136 (55%)	55	3,630	125	2.7	119	75
EFO-NCIT	2,507	1,731	1,541 (61%)	143	3,687	311	4.3	353	73
EP-FMA	78,489	348	109 (0.1%)	16	0	168	11.0	168	0
EP-NCIT	2,465	363	307 (12%)	69	253	136	3.8	180	0
MA-FMA	961	21	22 (2%)	1	0	1	2.0	2	0
OMIM-NCIT	5,178	1,800	1,078 (21%)	154	0	396	10.2	536	0
SDO-EP	135	3	3 (2%)	1	0	1	3.0	3	0
UBERON-FMA	1,932	486	121 (6%)	19	25	70	6.3	85	25
ZFA-EFO	427	7	11 (3%)	5	0	10	2.6	10	0
ZFA-UBERON	724	0	0 (0%)	0	104	0	0	0	104
Average	8,777	691	518 (22%)	94	700	122	5	146	25

$\mathcal{M}$: number of BioPortal mappings; $\lvert\mathcal{CS}\rvert$: number of conflict sets; $\lvert\mathcal{M}_{\mathcal{CS}}\rvert$: number of distinct mappings in conflict sets; $\lvert\mathcal{CG}\rvert$: number of conflict groups; $\overline{\mathcal{M}_{\mathcal{CG}}}$: average number of mappings per conflict group; $\lvert\mathcal{R}^{\approx}\rvert$: repair size in number of equivalence mappings (AML) or number of subsumption mappings (LogMap); **Unsat.**: unsatisfiable classes after repair.

The incoherence of the repaired mapping sets has been significantly reduced, and in many cases completely removed. The one exception was the ZFA-UBERON case, as neither AML-Repair nor LogMap-Repair could detect and repair any of the unsatisfiabilities in this alignment. Furthermore, the computed (approximate) repairs were not aggressive, as they removed at most 5.7% (AML-Repair) and 7% (LogMap-Repair) of the mappings (in the EFO-NCIT case).

In addition to producing a repair, AML-Repair also identifies the number of conflicting mapping sets $\mathcal{CS}$ and the total number of mappings that are involved in at least one conflict $\mathcal{M}_{\mathcal{CS}}$. For example, in the BDO-NCIT case, AML-Repair identifies 1,649 conflicting sets which involve 84% of the mappings $\mathcal{M}$ in this alignment. Given that these mappings were leading to 34,341 unsatisfiable classes (see Table 2), the fact that only 53 equivalence mappings were removed indicates that (at least) some of these were causing several unsatisfiabilities, likely because they were in conflict with multiple other mappings.

LogMap-Repair, on the other hand, identifies groups of potentially conflicting mappings $\mathcal{CG}$ (which contain one or more $\mathcal{CS}$) involved in each unsatisfiability, and the average number of mappings in each conflicting group $\overline{\mathcal{M}_{\mathcal{CG}}}$. $\mathcal{CG}$ represents a lower bound of the total number of groups, since LogMap-Repair repairs on-the-fly and removing one mapping may solve multiple unsatisfiabilities. For example, in the BDO-NCIT case, LogMap-Repair only identifies 125 conflicting groups with an average of 3.2 mappings per group. Note that solving the 125 unsatisfiabilities corresponding to the conflict groups is sufficient to repair the original 34,341 unsatisfiabilities, which again suggests that a few mappings in the conflict groups were causing many of these errors.

4.2 Manual Analysis

To complement the automatic repair evaluation and investigate the causes behind the incoherences identified therein, we analyzed manually the mappings removed by AML-Repair and LogMap-Repair (up to a maximum of 100 mappings per ontology pair, and in the case of LogMap-Repair, only the cases where the subsumption mappings were removed in both directions).

For each removed mapping, we assessed whether it was correct or erroneous (within the context of the ontologies). We deemed a mapping to be erroneous if it falls into one of the following categories:

1. At least one of the entities it maps is obsolete/retired, as in the mapping: BDO#HP_0001596 (Alopecia) $\Leftrightarrow$ NCIT#C2865 (Alopecia), where the latter class is retired in NCIT.
2. The entities it maps are not directly related, as in the mapping: BDO#PATO_0001901 (Back) $\Leftrightarrow$ NCIT#C13062 (Back), where the former class stands for the directional qualifier and the latter stands for the body part.
3. The entities it maps are related but the relationship between them should not be modeled as *skos:closeMatch*, as in the the mapping: BDO#G0000064 (CREBP) $\Leftrightarrow$ NCIT#C17803 (CREB-Binding Protein), which maps entities that are related (the gene and corresponding protein) but semantically distinct. Moreover, this mapping conflicts with the correct (protein-protein) mapping: BDO#P000022 (CREB-Binding Protein) $\Leftrightarrow$ NCIT#C17803 (CREB-Binding Protein).

Additionally, when the removed mapping was deemed to be correct, we analyzed the conflict sets $\mathcal{CS}$ in which the removed mapping was present (computed by AML-Repair, see Algorithm 3) and assessed whether the mappings in conflict with it were correct or erroneous. For the purpose of our evaluation, the main issue is not whether the repair algorithms remove erroneous mappings, but rather whether any of the unsatisfiabilities in which it is involved are caused by erroneous mappings. Thus, if either the removed mapping or at least one of its conflicting mappings was erroneous, we attributed the cause of removal to a mapping error. If the mapping itself and all of its conflicting mappings were correct, we considered the cause of removal to be an incompatibility between the ontologies.

The results of our manual analysis are summarized in Table 4. In total, over 40% of the mappings removed by both repair systems were indeed erroneous. Furthermore, errors in the mappings were the cause of removal of over 60% of the mappings.

We found that category 1 errors (i.e., mappings including obsolete/retired classes) were relatively common in all alignments that included NCIT. Furthermore, there were two common category 3 error patterns in these alignments: gene-protein matches, and human-mouse matches. The former pattern consists of a mapping between a gene and its corresponding protein or vice-versa, as exemplified above. The latter pattern consists of a mapping between a (Human) Health/Anatomy classes and a corresponding NCIT Mouse class (from the Mouse Pathologic Diagnoses or Mouse Anatomy Concepts sections) which naturally conflicts with the mapping to the main (Human) NCIT sections. One example of this pattern is the mapping: BDO#HP_0010786 (Urinary Tract Neoplasm) $\Leftrightarrow$ NCIT#C25806 (Mouse Urinary Tract Neoplasm). Another pattern of this

Table 4: Manual evaluation of the repaired BioPortal mappings

Ontology Pairs	**AML-Repair**			**LogMap-Repair**		
	Analyzed	**Erroneous**	**Err. Cause**	**Analyzed**	**Erroneous**	**Err. Cause**
BDO-NCIT	53	55%	83%	52	62%	83%
CCONT-NCIT	55	33%	62%	68	43%	59%
EFO-NCIT	100	53%	91%	100	54%	93%
EP-FMA	16	0%	0%	84	0%	0%
EP-NCIT	69	43%	71%	78	60%	73%
MA-FMA	1	100%	100%	1	100%	100%
OMIM-NCIT	100	48%	71%	100	49%	76%
SDO-EP	1	100%	100%	0	N/A	N/A
UBERON-FMA	19	0%	0%	20	0%	0%
ZFA-EFO	5	60%	100%	4	75%	100%
Total	419	44%	71%	507	42%	62%

category that occurred in the OMIM-NCIT alignment consists of a mapping between a disease/symptom and a corresponding adverse event, such as: OMIM#MTHU023845 (Neck Pain) ⇔ NCIT#C56135 (Neck Pain Adverse Event).

Regarding category 2 errors, there were few patterns other than number mismatches, such as in the mapping: BDO#G0000133 (TBX4) ⇔ NCIT#C101638 (TBX3 Gene). While the cause of these mismatches was often clear, as in the 'back' example above, some defy reason as the case of: EFO#CHEBI_15366 (Acetic Acid) ⇔ NCIT#C37392 (C58 Mouse).

As for incompatibilities between the ontologies, one of the most interesting cases is the EP-FMA alignment, which is actually an alignment between the OBO version of FMA (which is imported by EP) and the BioPortal version of FMA. Indeed, it was surprising to find that the alignment is incoherent, given that all mappings are true equivalences. It turns out that there are a few structural differences between the two versions of the ontology which cause the incoherences, as some entities are modeled as 'Material Anatomical Entity' in the OBO version and as 'Immaterial Anatomical Entity' in the BioPortal version (with the latter appearing to be more correct in most cases). The same type of structural differences is also behind the incoherences in the UBERON-FMA alignment. Also interesting is the OMIM-NCIT alignment, as OMIM models diseases as subclasses of the anatomical structures where they occur, whereas NCIT models diseases as disjoint from anatomical structures, making it impossible to obtain a coherent alignment between the ontologies where both diseases and anatomical structures are mapped.

4.3 Discussion

The results of our study reveal that many sets of BioPortal mappings lead to logical incoherences when taken as a whole, and that many of these incoherences involve erroneous mappings. Thus, adding annotations to BioPortal mappings about potential logical incompatibilities with other mappings would not only improve their usability for semantic web applications, which require logical integrity, but also contribute to identify and discard erroneous mappings.

Our study also demonstrates that approximate repair algorithms such as AML-Repair and LogMap-Repair can effectively identify most of the logical conflicts in BioPortal mappings, as well as the mappings that cause them. Furthermore, unlike complete repair algorithms such as those based on DDL [30], AML-Repair and LogMap-Repair are feasible in practice (repair times in AML-Repair ranged from 10 seconds to 10 minutes, whereas in LogMap-Repair they ranged from 3 to 92 seconds, in a quad-core computer with 8 GB allocated RAM). Thus these algorithms could play a pivotal role in the task of identifying and annotating conflicts between BioPortal mappings.

Furthermore, in addition to the annotation of existing mappings, AML-Repair and LogMap-Repair could be employed to screen newly submitted mappings, so that those leading to logical conflicts can be reviewed before being integrated into BioPortal. This could effectively preclude the addition of some erroneous mappings, and would enable the immediate annotation of the mappings accepted for integration.

Regarding the categories of erroneous mappings found, the category 1 errors (i.e, mappings that include retired/obsolete entities) are straightforward to identify and handle automatically, even without the use of repair algorithms. These mappings should not be maintained as 'active' mappings in BioPortal given that retired/obsolete entities are no longer be considered an active part of the ontologies. However, it makes sense to keep track of such mappings, so the best solution would be to annotate the mappings themselves as obsolete.

Category 2 errors (i.e., mappings between unrelated classes) should definitely be excluded from BioPortal, whereas category 3 errors (i.e., mappings between classes that are related but not closely) can be addressed either by exclusion or by changing the semantic relationship. For instance, a gene and its corresponding protein could be considered *skos:relatedMatch* rather than *skos:closeMatch*, although ideally a more descriptive mapping relation should be used. However, if both ontologies describe the gene and the protein, at least one ontology describes the relation between them, and BioPortal includes both the gene-gene and the protein-protein mappings; then maintaining a gene-protein mapping is semantically redundant.

Although finding category 2 and 3 errors typically requires human intervention, the use of mapping repair algorithms is critical to facilitate their detection. Note that, not all erroneous mappings necessarily lead to logical conflicts, particularly when the ontologies lack disjointness definitions. Nevertheless, addressing conflict-causing errors will surely be a significant improvement, and the common error patterns thus identified can be employed to search for (non-conflict causing) errors, even in ontologies that lack disjointness restrictions.

The identification of logical conflicts caused by inherent incompatibilities between the ontologies is also critical to understand the limits of interoperability. For instance, integrating OMIM and NCIT requires excluding either mappings between anatomic entities or mappings between diseases (depending on the intended application), or ultimately relaxing the disjointness restrictions in the NCIT. Additionally, such incompatibilities may point to modeling errors in the ontologies, as in the EP-FMA case, and enable their correction.

5 Conclusions

BioPortal fulfills a critical need of the biomedical community by promoting integration and interoperability between the numerous biomedical ontologies with overlapping domains. However, the different scopes of these ontologies often lead to incompatible views of a given domain, placing restrictions on interoperability. Maintaining conflicting mappings may best serve the needs of the community, as a wider mapping coverage will satisfy more users and enable more applications. Nevertheless, if BioPortal mappings are to be usable on a large scale, and particularly by automatic applications, then identifying those that lead to logical errors is paramount.

Another issue that affects BioPortal are mapping errors, which are inevitable on this scale, particularly when most mappings are produced by automated ontology matching techniques. Finding and correcting these errors is a daunting task, but one of the utmost importance, as they are likely to propagate if used to draw inferences. While not all errors cause logical conflicts, many do, and as our evaluation illustrates, identifying these enables the discovery of error patterns that can be applied to identify further errors.

Identifying logical conflicts in BioPortal mappings thus serves the dual purpose of improving usability and facilitating error detection. Given that using complete reasoners for this task is unfeasible, due to the scale of BioPortal, approximate mapping repair systems such as AML-Repair and LogMap-Repair appear to be the ideal solution. Indeed, our study has shown that these systems are both effective and efficient in tackling large sets of mappings, and will be even more efficient considering that the goal is only to identify conflicts rather than to repair them.

Our proposal is that BioPortal mappings be enriched with annotations about other mappings they conflict with, a solution which fits into BioPortal's community-driven and multiple-perspective approach. While distinguishing mappings errors from incompatibilities will require manual analysis, this is a task that could be carried out gradually by the community once mappings are annotated with logical conflicts.

There is, however, one type of error that can be addressed immediately: mappings that include obsolete/retired entities. Despite the fact that there are different representations of these entities among BioPortal ontologies, identifying them (and their mappings) should be straightforward to do automatically. We propose that such mappings be annotated as obsolete, which would enable BioPortal and its users to keep track of them while allowing their automatic exclusion by applications.

Our next step will be to contact BioPortal developers and collaborate with them in the process of finding and annotating mappings with information about logical conflicts, by applying our repair algorithms to the whole BioPortal.

Acknowledgements

This work was supported by the EU FP7 IP project Optique (no. 318338), the EPSRC project Score!, the Portuguese FCT through the SOMER project (PTDC/

EIA-EIA/119119/2010), and the LASIGE Strategic Project (PEst-OE/EEI/UI0408/ 2014).

We would like to thank the invaluable help provided by Bernardo Cuenca Grau and Ian Horrocks in the development of LogMap-Repair, and by Isabel F. Cruz in the development of AML-Repair.

References

1. Ashburner, M., Ball, C.A., Blake, J.A., Botstein, D., Butler, H., Cherry, J.M., Davis, A.P., Dolinski, K., Dwight, S.S., Eppig, J.T., et al.: Gene Ontology: tool for the unification of biology. Nature genetics 25(1), 25–29 (2000)
2. Bodenreider, O.: The unified medical language system (UMLS): integrating biomedical terminology. Nucleic Acids Research 32, 267–270 (2004)
3. Borgida, A., Serafini, L.: Distributed description logics: Assimilating information from peer sources. J. Data Sem. 1, 153–184 (2003)
4. Cuenca Grau, B., Horrocks, I., Motik, B., Parsia, B., Patel-Schneider, P.F., Sattler, U.: OWL 2: The next step for OWL. J. Web Sem. 6(4), 309–322 (2008)
5. Dowling, W.F., Gallier, J.H.: Linear-time algorithms for testing the satisfiability of propositional Horn formulae. J. Log. Prog. 1(3), 267–284 (1984)
6. Euzenat, J.: Semantic precision and recall for ontology alignment evaluation. In: Int'l Joint Conf. on Artif. Intell. (IJCAI). pp. 348–353 (2007)
7. Euzenat, J., Meilicke, C., Stuckenschmidt, H., Shvaiko, P., Trojahn, C.: Ontology alignment evaluation initiative: Six years of experience. J. Data Sem. 15, 158–192 (2011)
8. Euzenat, J., Shvaiko, P.: Ontology matching. Springer (2007)
9. Faria, D., Pesquita, C., Santos, E., Palmonari, M., Cruz, I.F., Couto, F.M.: The agreementmakerlight ontology matching system. In: OTM Conferences. pp. 527–541 (2013)
10. Fridman Noy, N., Shah, N.H., Whetzel, P.L., Dai, B., Dorf, M., Griffith, N., Jonquet, C., Rubin, D.L., Storey, M.A.D., Chute, C.G., Musen, M.A.: BioPortal: ontologies and integrated data resources at the click of a mouse. Nucleic Acids Research 37(Web-Server-Issue) (2009)
11. Gallo, G., Urbani, G.: Algorithms for testing the satisfiability of propositional formulae. J. Log. Prog. 7(1), 45–61 (1989)
12. Ghazvinian, A., Noy, N.F., Jonquet, C., Shah, N.H., Musen, M.A.: What four million mappings can tell you about two hundred ontologies. In: Int'l Sem. Web Conf. (ISWC) (2009)
13. Ghazvinian, A., Noy, N.F., Musen, M.A.: Creating mappings for ontologies in biomedicine: Simple methods work. In: AMIA Annual Symposium (AMIA) (2009)
14. Golbeck, J., Fragoso, G., Hartel, F.W., Hendler, J.A., Oberthaler, J., Parsia, B.: The National Cancer Institute's Thésaurus and Ontology. J. Web Sem. 1(1), 75–80 (2003)
15. Golbreich, C., Horridge, M., Horrocks, I., Motik, B., Shearer, R.: OBO and OWL: Leveraging Semantic Web Technologies for the Life Sciences. In: Int'l Sem. Web Conf. (2007)
16. Jiménez-Ruiz, E., Cuenca Grau, B., Horrocks, I.: On the feasibility of using OWL 2 DL reasoners for ontology matching problems. In: OWL Reasoner Evaluation Workshop (2012)
17. Jiménez-Ruiz, E., Cuenca Grau, B.: LogMap: Logic-based and Scalable Ontology Matching. In: Int'l Sem. Web Conf. (ISWC). pp. 273–288 (2011)
18. Jiménez-Ruiz, E., Cuenca Grau, B., Horrocks, I., Berlanga, R.: Ontology integration using mappings: Towards getting the right logical consequences. In: Eur. Sem. Web Conf. (2009)
19. Jiménez-Ruiz, E., Cuenca Grau, B., Horrocks, I., Berlanga, R.: Logic-based Assessment of the Compatibility of UMLS Ontology Sources. J. Biomed. Semant. 2(Suppl 1), S2 (2011)
20. Jiménez-Ruiz, E., Cuenca Grau, B., Zhou, Y., Horrocks, I.: Large-scale interactive ontology matching: Algorithms and implementation. In: Europ. Conf. on Artif. Intell. (ECAI) (2012)

21. Jiménez-Ruiz, E., Meilicke, C., Grau, B.C., Horrocks, I.: Evaluating mapping repair systems with large biomedical ontologies. In: Description Logics. pp. 246–257 (2013)
22. Kalyanpur, A., Parsia, B., Horridge, M., Sirin, E.: Finding all justifications of OWL DL entailments. In: Int'l Sem. Web Conf. (ISWC). pp. 267–280. Springer (2007)
23. Kazakov, Y., Krötzsch, M., Simancik, F.: Concurrent classification of EL ontologies. In: Int'l Sem. Web Conf. (ISWC). pp. 305–320 (2011)
24. Konev, B., Walther, D., Wolter, F.: The Logical Difference Problem for Description Logic Terminologies. In: Int'l Joint Conf. on Automated Reasoning (IJCAR). pp. 259–274 (2008)
25. Meilicke, C.: Alignments Incoherency in Ontology Matching. Ph.D. thesis, University of Mannheim (2011)
26. Meilicke, C., Stuckenschmidt, H.: Incoherence as a basis for measuring the quality of ontology mappings. In: Ontology Matching Workshop (2008)
27. Meilicke, C., Stuckenschmidt, H., Tamilin, A.: Repairing ontology mappings. In: Proc. of AAAI Conf. on Artif. Intell. pp. 1408–1413 (2007)
28. Meilicke, C., Stuckenschmidt, H., Tamilin, A.: Reasoning support for mapping revision. J. Log. Comput. 19(5) (2009)
29. Noy, N.F., Griffith, N., Musen, M.A.: Collecting community-based mappings in an ontology repository. In: International Semantic Web Conference (ISWC). pp. 371–386 (2008)
30. Pathak, J., Chute, C.G.: Debugging Mappings between Biomedical Ontologies: Preliminary Results from the NCBO BioPortal Mapping Repository. In: Int'l Conf. on Biomedical Ontology (ICBO) (2009)
31. Pesquita, C., Faria, D., Santos, E., Couto, F.M.: To repair or not to repair: reconciling correctness and coherence in ontology reference alignments. In: Ontology Matching (OM) (2013)
32. Rosse, C., Mejino Jr., J.: A reference ontology for biomedical informatics: the Foundational Model of Anatomy. J. Biomed. Informatics 36(6), 478–500 (2003)
33. Salvadores, M., Alexander, P.R., Musen, M.A., Noy, N.F.: BioPortal as a dataset of linked biomedical ontologies and terminologies in RDF. Semantic Web 4(3), 277–284 (2013)
34. Santos, E., Faria, D., Pesquita, C., Couto, F.: Ontology alignment repair through modularization and confidence-based heuristics. arXiv:1307.5322 preprint (2013)
35. Santos, E., Faria, D., Pesquita, C., Couto, F.M.: Ontology alignment repair through modularization and confidence-based heuristics. CoRR abs/1307.5322 (2013)
36. Schlobach, S.: Debugging and semantic clarification by pinpointing. In: The Semantic Web: Research and Applications, pp. 226–240. Springer (2005)
37. Shvaiko, P., Euzenat, J.: Ontology matching: State of the art and future challenges. IEEE Trans. Knowledge and Data Eng. (2012)
38. Smith, B., Ashburner, M., Rosse, C., Bard, J., Bug, W., Ceusters, W., Goldberg, L.J., Eilbeck, K., Ireland, A., Mungall, C.J., Leontis, N., Rocca-Serra, P., Ruttenberg, A., Sansone, S.A., Scheuermann, R.H., Shah, N., Whetzel, P.L., Lewis, S.: The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration. Nat Biotech 25(11) (2007)
39. Suntisrivaraporn, B., Qi, G., Ji, Q., Haase, P.: A modularization-based approach to finding all justifications for OWL DL entailments. In: Asian Sem. Web Conf. (ASWC) (2008)

Appendix H

ISWC 2014: Ontology Approximation

This appendix reports the paper:

– Marco Console, Jose Mora, Riccardo Rosati, Valerio Santarelli, Domenico Fabio Savo, Effective computation of maximal sound approximations of Description Logic ontologies. In Proceedings of the International Semantic Web Conference 2014

Effective computation of maximal sound approximations of Description Logic ontologies

Marco Console, Jose Mora, Riccardo Rosati, Valerio Santarelli, Domenico Fabio Savo

Dipartimento di Ingegneria Informatica Automatica e Gestionale Antonio Ruberti
Sapienza Università di Roma
lastname@dis.uniroma1.it

Abstract. We study the problem of approximating Description Logic (DL) ontologies specified in a source language $\mathcal{L}_S$ in terms of a less expressive target language $\mathcal{L}_T$. This problem is getting very relevant in practice: e.g., approximation is often needed in ontology-based data access systems, which are able to deal with ontology languages of a limited expressiveness. We first provide a general, parametric, and semantically well-founded definition of maximal sound approximation of a DL ontology. Then, we present an algorithm that is able to effectively compute two different notions of maximal sound approximation according to the above parametric semantics when the source ontology language is OWL 2 and the target ontology language is OWL 2 QL. Finally, we experiment the above algorithm by computing the two OWL 2 QL approximations of a large set of existing OWL 2 ontologies. The experimental results allow us both to evaluate the effectiveness of the proposed notions of approximation and to compare the two different notions of approximation in real cases.

1 Introduction

Description Logic (DL) ontologies are the core element of ontology-based data access (OBDA) [15], in which the ontology is utilized as a conceptual view, allowing user access to the underlying data sources. In OBDA, as well as in all the current applications of ontologies requiring automated reasoning, a trade-off between the expressiveness of the ontology specification language and the complexity of reasoning in such a language must be reached. More precisely, most of the current research and development in OBDA is focusing on languages for which reasoning, and in particular query answering, is not only tractable (in data complexity) but also *first-order* rewritable [2,5]. This imposes significant limitations on the set of constructs and axioms allowed in the ontology language.

The limited expressiveness of the current ontology languages adopted in OBDA provides a strong motivation for studying the approximation of ontologies formulated in (very) expressive languages with ontologies in low-complexity languages such as OWL 2 QL. Such a motivation is not only theoretical, but also practical, given the current availability of OBDA systems and the increasing interest in applying the OBDA approach in the real world [1,6,7,16]: for instance, ontology approximation is currently one of the main issues in the generation of ontologies for OBDA within the use cases of the Optique EU project.[1]

Several approaches have recently dealt with the problem of approximating Description Logic ontologies. These can roughly be partitioned in two types: *syntactic* and *semantic*. In the former, only the syntactic form of the axioms of the original ontology is considered, thus those axioms which do not comply with the syntax of the target ontology language are disregarded [17,18]. This approach generally can be performed quickly and through simple algorithms. However, it does not, in general, guarantee soundness, i.e., to infer only correct entailments, or completeness, i.e., all entailments of the original ontology that are also expressible in the target language are preserved [14]. In the latter, the object of the approximation are the entailments of the original ontology, and the goal is to preserve as much as possible of these entailments by means of an ontology in the target language, guaranteeing soundness of the result. On the other hand, this approach often necessitates to perform complex reasoning tasks over the ontology, possibly resulting significantly slower. For these reasons, the semantic approach to ontology approximation poses a more interesting but more complex challenge.

In this paper, we study the problem of approximating DL ontologies specified in a source language $\mathcal{L}_s$ in terms of a less expressive target language $\mathcal{L}_t$. We deal with this problem by first providing a general, parametric, and semantically well-founded definition of maximal sound approximation of a DL ontology. Our semantic definition captures and generalizes previous approaches to ontology approximation [4,8,11,14]. In particular, our approach builds on the preliminary work presented in [8], which proposed a similar, although non-parameterized, notion of maximal sound approximation.

Then, we present an algorithm that is able to effectively compute two different notions of maximal sound approximation according to the above parametric semantics, when the source ontology language is OWL 2 and the target ontology language is OWL 2 QL. In particular, we focus on the *local semantic approximation (LSA)* and the *global semantic approximation (GSA)* of a source ontology. These two notions of approximation correspond to the cases when the parameter of our semantics is set, respectively, to its minimum and to its maximum. Informally, the LSA of an ontology is obtained by considering (and reasoning over) one axiom α of the source ontology at a time, so this technique tries to approximate α independently of the rest of the source ontology. On the contrary, the GSA tries to approximate the source ontology by considering all its axioms (and reasoning over such axioms) at the same time. As a consequence, the GSA is potentially able to "approximate better" than the LSA, while the LSA appears in principle computationally less expensive than the GSA. Notably, in the case of OWL 2 QL, the GSA corresponds to the notion of approximation given in [14], which has been shown to be very well-suited for query answering purposes.

Finally, we experiment the above algorithm by computing the LSA and the GSA in OWL 2 QL of a large set of existing OWL 2 ontologies. The experimental results allow us both to evaluate the effectiveness of the proposed notions of approximation and to compare the two different notions of approximation in real cases. In particular, the main properties pointed out by our experimental results are the following:

[1] http://optique-project.eu

1. the computation of the LSA is usually less expensive than computing the GSA of a given source ontology;
2. in many cases, both the LSA and the GSA of an ontology are very good approximations of the ontology, in the sense that the approximated ontologies actually entail a large percentage of the axioms of the source ontology;
3. in many cases, the LSA and the GSA coincide. This and the previous property imply that the computationally less expensive LSA is usually already able to compute a high-quality sound approximation of the source ontology.

The paper is structured in the following way. First, in Section 2 we recall DL ontology languages, in particular OWL 2 and OWL 2 QL. Then, in Section 3 we present our formal, parameterized notion of semantic sound approximation of an ontology, and illustrate some general properties of such a notion. In Section 4 we present the techniques for computing the GSA and the LSA of OWL 2 ontologies in OWL 2 QL. Finally, we present an experimental evaluation of the above techniques in Section 5, and draw some conclusions in Section 6.

2 Preliminaries

Description Logics (DLs) [3] are logics that allow one to represent the domain of interest in terms of *concepts*, denoting sets of objects, *value-domains*, denoting sets of values, *attributes*, denoting binary relations between objects and values, and *roles* denoting binary relations over objects.

In this paper we consider the DL $\mathcal{SROIQ}$ [10], which is the logic underpinning OWL 2, as the "maximal" DL considered in this paper.

Let Σ be a signature of symbols for individual (object and value) constants and predicates, i.e., concepts, value-domains, attributes, and roles. Let Φ be the set of all $\mathcal{SROIQ}$ axioms over Σ.

An *ontology* over Σ is a finite subset of Φ.

A *DL language* over Σ (or simply *language*) $\mathcal{L}$ is a set of ontologies over Σ. We call *$\mathcal{L}$-ontology* any ontology $\mathcal{O}$ such that $\mathcal{O} \in \mathcal{L}$. Moreover, we denote by $\Phi_{\mathcal{L}}$ the set of axioms $\bigcup_{\mathcal{O} \in \mathcal{L}} \mathcal{O}$.

We call a language $\mathcal{L}$ *closed* if $\mathcal{L} = 2^{\Phi_{\mathcal{L}}}$. As we will see in the following, there exist both closed and non-closed DL languages among the standard ones.

The semantics of an ontology is given in terms of first-order (FOL) interpretations (cf. [3]). We denote with $Mod(\mathcal{O})$ the set of models of $\mathcal{O}$, i.e., the set of FOL interpretations that satisfy all the axioms in $\mathcal{O}$ (we recall that every $\mathcal{SROIQ}$ axiom corresponds to a first-order sentence). As usual, an ontology $\mathcal{O}$ is said to be *satisfiable* if it admits at least one model, and $\mathcal{O}$ is said to *entail* a First-Order Logic (FOL) sentence α, denoted $\mathcal{O} \models \alpha$, if $\alpha^{\mathcal{I}} = \mathit{true}$ for all $\mathcal{I} \in Mod(\mathcal{O})$. Moreover, given two ontologies $\mathcal{O}$ and $\mathcal{O}'$, we say that $\mathcal{O}$ and $\mathcal{O}'$ are logically equivalent if $Mod(\mathcal{O}) = Mod(\mathcal{O}')$.

In this work we will mainly focus on two specific languages, which are OWL 2, the official ontology language of the World Wide Web Consortium (W3C) [9], and one of its profiles, OWL 2 QL [12]. Due to the limitation of space, here we do not provide a complete description of OWL 2, and refer the reader to the official W3C specification [13].

We now present the syntax of OWL 2 QL. We use the German notation for describing OWL 2 QL constructs and axioms, and refer the reader to [12] for the OWL functional style syntax.

Expressions in OWL 2 QL are formed according to the following syntax:

$$\begin{array}{ll} B \longrightarrow A \mid \exists Q \mid \delta_F(U) \mid \top_C \mid \bot_C & E \longrightarrow \rho(U) \\ C \longrightarrow B \mid \neg B \mid \exists Q.A & F \longrightarrow \top_D \mid T_1 \mid \cdots \mid T_n \\ Q \longrightarrow P \mid P^- \mid \top_P \mid \bot_P & V \longrightarrow U \mid \top_A \mid \bot_A \\ R \longrightarrow Q \mid \neg Q & W \longrightarrow V \mid \neg V \end{array}$$

where: A, P, and U are symbols denoting respectively an *atomic concept*, an *atomic role*, and an *atomic attribute*; P^- denotes the inverse of P; $\exists Q$, also called *unqualified existential role*, denotes the set of objects related to some object by the role Q; $\delta_F(U)$ denotes the *qualified domain* of U with respect to a value-domain F, i.e., the set of objects that U relates to some value in F; $\rho(U)$ denotes the *range* of U, i.e., the set of values related to objects by U; $T_1, \ldots, T_n$ denote n unbounded value-domains (i.e., datatypes); the concept $\exists Q.A$, or *qualified existential role*, denotes the *qualified domain* of Q with respect to A, i.e., the set of objects that Q relates to some instance of A. $\top_C$, $\top_P$, $\top_A$, and $\top_D$ denote, respectively, the universal concept, role, attribute, and value-domain, while $\bot_C$, $\bot_P$, and $\bot_A$ denote, respectively, the empty concept, role, and attribute.

An OWL 2 QL ontology $\mathcal{O}$ is a finite set of axioms of the form:

$$B \sqsubseteq C \qquad Q \sqsubseteq R \qquad U \sqsubseteq V \qquad E \sqsubseteq F \qquad ref(P) \qquad irref(P)$$
$$A(a) \qquad P(a,b) \qquad U(a,v)$$

From left to right, the first four above axioms denote subsumptions between concepts, roles, attributes, and value-domains, respectively. The fifth and sixth axioms denote reflexivity and irreflexivity on roles. The last three axioms denote membership of an individual to a concept, membership of a pair of individuals to a role, and membership of a pair constituted by an individual and a value to an attribute.

From the above definition, it immediately follows that OWL 2 QL is a closed language. On the other hand, we recall that OWL 2 is not a closed language. This is due to the fact that OWL 2 imposes syntactic restrictions that concern the simultaneous presence of multiple axioms in the ontology (for instance, there exist restrictions on the usage of role names appearing in role inclusions in the presence of the role chaining constructor).

3 Approximation

In this section, we illustrate our notion of approximation in a target language $\mathcal{L}_T$ of an ontology $\mathcal{O}_S$ in a language $\mathcal{L}_S$.

Typically, when discussing approximation, one of the desirable properties is that of soundness. Roughly speaking, when the object of approximation is a set of models, this property requires that the set of models of the approximation is a superset of those of the original ontology. Another coveted characteristic of the computed ontology is that it be the "best" approximation of the original ontology. In other words, the need of keeping a minimal distance between the original ontology and the ontology resulting from its approximation is commonly perceived.

On the basis of these observations, the following definition of approximation in a target language $\mathcal{L}_T$ of a satisfiable $\mathcal{L}_S$-ontology is very natural.

Definition 1. *Let $\mathcal{O}_S$ be a satisfiable $\mathcal{L}_S$-ontology, and let $\Sigma_{\mathcal{O}_S}$ be the set of predicate and constant symbols occurring in $\mathcal{O}_S$. An $\mathcal{L}_T$-ontology $\mathcal{O}_T$ over $\Sigma_{\mathcal{O}_S}$ is a* global semantic approximation *(GSA) in $\mathcal{L}_T$ of $\mathcal{O}_S$ if both the following statements hold:*

(i) $Mod(\mathcal{O}_S) \subseteq Mod(\mathcal{O}_T)$*;*
(ii) there is no $\mathcal{L}_T$-ontology $\mathcal{O}'$ over $\Sigma_{\mathcal{O}_S}$ such that $Mod(\mathcal{O}_S) \subseteq Mod(\mathcal{O}') \subset Mod(\mathcal{O}_T)$.

We denote with $globalApx(\mathcal{O}_S, \mathcal{L}_T)$ the set of all the GSAs in $\mathcal{L}_T$ of $\mathcal{O}_S$.

In the above definition, statement (i) imposes the soundness of the approximation, while statement (ii) imposes the condition of “closeness” in the choice of the approximation.

We observe that an $\mathcal{L}_T$-ontology which is the GSA in $\mathcal{L}_T$ of $\mathcal{O}_S$ may not exist. This is the case when, for each $\mathcal{L}_T$ ontology $\mathcal{O}'_T$ satisfying statement (i) of Definition 1, there always exists an $\mathcal{L}_T$-ontology $\mathcal{O}''_T$ which satisfies statement (i), but for which we have that $Mod(\mathcal{O}_S) \subseteq Mod(\mathcal{O}''_T) \subset Mod(\mathcal{O}'_T)$.

The following lemma provides a sufficient condition for the existence of the GSA in a language $\mathcal{L}_T$ of an ontology $\mathcal{O}_S$.

Lemma 1. *Given a language $\mathcal{L}_T$ and a finite signature Σ, if the set of non-equivalent axioms in $\Phi_{\mathcal{L}_T}$ that one can generate over Σ is finite, then for any $\mathcal{L}_S$-ontology $\mathcal{O}_S$ $globalApx(\mathcal{O}_S, \mathcal{L}_T) \neq \emptyset$.*

In cases where GSAs exist, i.e., $globalApx(\mathcal{O}_S, \mathcal{L}_T) \neq \emptyset$, given two ontologies $\mathcal{O}'$ and $\mathcal{O}''$ in $globalApx(\mathcal{O}_S, \mathcal{L}_T)$, they may be either logically equivalent or not. The condition of non-equivalence is due to the fact that the language in which the original ontology is approximated is not closed. We have the following lemma.

Lemma 2. *Let $\mathcal{L}_T$ be a closed language, and let $\mathcal{O}_S$ be an ontology. For each $\mathcal{O}'$ and $\mathcal{O}''$ belonging to $globalApx(\mathcal{O}_S, \mathcal{L}_T)$, we have that $\mathcal{O}'$ and $\mathcal{O}''$ are logically equivalent.*

Proof. Towards a contradiction, suppose that $Mod(\mathcal{O}') \neq Mod(\mathcal{O}'')$. From this, and from Definition 1 we have that $Mod(\mathcal{O}') \not\subset Mod(\mathcal{O}'')$ and $Mod(\mathcal{O}'') \not\subset Mod(\mathcal{O}')$. Hence, there exist axioms α and β in $\Phi_{\mathcal{L}_T}$ such that $\mathcal{O}' \models \alpha$ and $\mathcal{O}'' \not\models \alpha$, and $\mathcal{O}'' \models \beta$ and $\mathcal{O}' \not\models \beta$. Since both $\mathcal{O}'$ and $\mathcal{O}''$ are sound approximations of $\mathcal{O}_S$, $\mathcal{O}_S \models \{\alpha, \beta\}$. Because $\mathcal{L}_T$ is closed, the ontology $\mathcal{O}'_\beta = \mathcal{O}' \cup \{\beta\}$ is an $\mathcal{L}_T$-ontology. From the above considerations it directly follows that $Mod(\mathcal{O}_S) \subseteq Mod(\mathcal{O}'_\beta) \subset Mod(\mathcal{O}')$. This means that $\mathcal{O}'$ does not satisfy condition (ii) of Definition 1, and therefore $\mathcal{O}' \notin globalApx(\mathcal{O}_S, \mathcal{L}_T)$, which is a contradiction. The same conclusion can be reached analogously for $\mathcal{O}''$. ■

In other words, if the target language is closed, Lemma 2 guarantees that, up to logical equivalence, the GSA is unique.

Definition 1 is non-constructive, in the sense that it does not provide any hint as to how to compute the approximation in $\mathcal{L}_T$ of an ontology $\mathcal{O}_S$. The following theorem suggests more constructive conditions, equivalent to those in Definition 1, but first we need to introduce the notion of *entailment set* [14] of a satisfiable ontology with respect to a language.

Definition 2. *Let $\Sigma_{\mathcal{O}}$ be the set of predicate and constant symbols occurring in $\mathcal{O}$, and let $\mathcal{L}'$ be a language. The* entailment set *of $\mathcal{O}$ with respect to $\mathcal{L}'$, denoted as $\mathsf{ES}(\mathcal{O}, \mathcal{L}')$, is the set of axioms from $\Phi_{\mathcal{L}'}$ that only contain predicates and constant symbols from $\Sigma_{\mathcal{O}}$ and that are entailed by $\mathcal{O}$.*

In other words, we say that an axiom α belongs to the entailment set of an ontology $\mathcal{O}$ with respect to a language $\mathcal{L}'$, if α is an axiom in $\Phi_{\mathcal{L}'}$ built over the signature of $\mathcal{O}$ and for each interpretation $\mathcal{I} \in Mod(\mathcal{O})$ we have that $\mathcal{I} \models \alpha$.

Clearly, given an ontology $\mathcal{O}$ and a language $\mathcal{L}'$, the entailment set of $\mathcal{O}$ with respect to $\mathcal{L}'$ is unique.

Theorem 1. *Let $\mathcal{O}_S$ be a satisfiable $\mathcal{L}_S$-ontology and let $\mathcal{O}_T$ be a satisfiable $\mathcal{L}_T$-ontology. We have that:*

(a) $Mod(\mathcal{O}_S) \subseteq Mod(\mathcal{O}_T)$ if and only if $\mathsf{ES}(\mathcal{O}_T, \mathcal{L}_T) \subseteq \mathsf{ES}(\mathcal{O}_S, \mathcal{L}_T)$;
(b) there is no $\mathcal{L}_T$-ontology $\mathcal{O}'$ such that $Mod(\mathcal{O}_S) \subseteq Mod(\mathcal{O}') \subset Mod(\mathcal{O}_T)$ if and only if there is no $\mathcal{L}_T$-ontology $\mathcal{O}''$ such that $\mathsf{ES}(\mathcal{O}_T, \mathcal{L}_T) \subset \mathsf{ES}(\mathcal{O}'', \mathcal{L}_T) \subseteq \mathsf{ES}(\mathcal{O}_S, \mathcal{L}_T)$.

Proof. We start by focusing on the first statement. ($\Leftarrow$) Suppose, by way of contradiction, that $\mathsf{ES}(\mathcal{O}_T, \mathcal{L}_T) \subseteq \mathsf{ES}(\mathcal{O}_S, \mathcal{L}_T)$ and that $Mod(\mathcal{O}_S) \not\subseteq Mod(\mathcal{O}_T)$. This means that there exists at least one interpretation that is a model for $\mathcal{O}_S$ but not for $\mathcal{O}_T$. Therefore there exists an axiom $\alpha \in \mathcal{O}_T$ such that $\mathcal{O}_S \not\models \alpha$. Since $\mathcal{O}_T$ is an ontology in $\mathcal{L}_T$, then α is an axiom in $\Phi_{\mathcal{L}_T}$. It follows that $\alpha \in \mathsf{ES}(\mathcal{O}_T, \mathcal{L}_T)$ and that $\alpha \notin \mathsf{ES}(\mathcal{O}_S, \mathcal{L}_T)$, which leads to a contradiction.

($\Rightarrow$) Towards a contradiction, suppose that $Mod(\mathcal{O}_S) \subseteq Mod(\mathcal{O}_T)$, but $\mathsf{ES}(\mathcal{O}_T, \mathcal{L}_T) \not\subseteq \mathsf{ES}(\mathcal{O}_S, \mathcal{L}_T)$. This means that there exists at least one axiom $\alpha \in \mathsf{ES}(\mathcal{O}_T, \mathcal{L}_T)$ such that $\alpha \notin \mathsf{ES}(\mathcal{O}_S, \mathcal{L}_T)$. It follows that $\mathcal{O}_T \models \alpha$ while $\mathcal{O}_S \not\models \alpha$, which immediately implies that $Mod(\mathcal{O}_S) \not\subseteq Mod(\mathcal{O}_T)$. Hence we have a contradiction.

Now we prove the second statement. ($\Leftarrow$) By contradiction, suppose that there exists an $\mathcal{L}_T$-ontology $\mathcal{O}'$ such that $Mod(\mathcal{O}_S) \subseteq Mod(\mathcal{O}') \subset Mod(\mathcal{O}_T)$, and that there does not exist any $\mathcal{L}_T$-ontology $\mathcal{O}''$ such that $\mathsf{ES}(\mathcal{O}_T, \mathcal{L}_T) \subset \mathsf{ES}(\mathcal{O}'', \mathcal{L}_T) \subseteq \mathsf{ES}(\mathcal{O}_S, \mathcal{L}_T)$. From what shown before, we have that $Mod(\mathcal{O}_S) \subseteq Mod(\mathcal{O}') \subseteq Mod(\mathcal{O}_T)$ implies that $\mathsf{ES}(\mathcal{O}_T, \mathcal{L}_T) \subseteq \mathsf{ES}(\mathcal{O}', \mathcal{L}_T) \subseteq \mathsf{ES}(\mathcal{O}_S, \mathcal{L}_T)$. Moreover, since both $\mathcal{O}'$ and $\mathcal{O}_T$ are $\mathcal{L}_T$ ontologies, $Mod(\mathcal{O}') \subset Mod(\mathcal{O}_T)$ implies that $\mathsf{ES}(\mathcal{O}_T, \mathcal{L}_T) \neq \mathsf{ES}(\mathcal{O}', \mathcal{L}_T)$. Hence, $\mathsf{ES}(\mathcal{O}_T, \mathcal{L}_T) \subset \mathsf{ES}(\mathcal{O}', \mathcal{L}_T) \subseteq \mathsf{ES}(\mathcal{O}_S, \mathcal{L}_T)$, which contradicts the hypothesis.

($\Rightarrow$) Suppose, by way of contradiction, that there exists an $\mathcal{L}_T$-ontology $\mathcal{O}''$ such that $\mathsf{ES}(\mathcal{O}_T, \mathcal{L}_T) \subset \mathsf{ES}(\mathcal{O}'', \mathcal{L}_T) \subseteq \mathsf{ES}(\mathcal{O}_S, \mathcal{L}_T)$ and there is no $\mathcal{L}_T$-ontology $\mathcal{O}'$

such that $Mod(\mathcal{O}_S) \subseteq Mod(\mathcal{O}') \subset Mod(\mathcal{O}_T)$. From property (a) we have that $Mod(\mathcal{O}_S) \subseteq Mod(\mathcal{O}'') \subseteq Mod(\mathcal{O}_T)$. Since both $\mathcal{O}''$ and $\mathcal{O}_T$ are $\mathcal{L}_T$ ontologies, then $\mathsf{ES}(\mathcal{O}_T, \mathcal{L}_T) \subset \mathsf{ES}(\mathcal{O}'', \mathcal{L}_T)$ implies that $Mod(\mathcal{O}'') \neq Mod(\mathcal{O}_T)$, which directly leads to a contradiction. ■

From Theorem 1 it follows that every ontology $\mathcal{O}_T$ which is a GSA in $\mathcal{L}_T$ of an ontology $\mathcal{O}_S$ is also an approximation in $\mathcal{L}_T$ of $\mathcal{O}_S$ according to [8], and, as we shall show in the following section, for some languages, this corresponds to the approximation in [14].

As discussed in [8], the computation of a GSA can be a very challenging task even when approximating into tractable fragments of OWL 2 [12]. This means that even though a GSA is one that best preserves the semantics of the original ontology, it currently suffers from a significant practical setback: the outcome of the computation of the approximation is tightly linked to the capabilities of the currently available reasoners for $\mathcal{L}_S$-ontologies. This may lead, in practice, to the impossibility of computing GSAs of very large or complex ontologies when the source language is very expressive.

We observe that the critical point behind these practical difficulties in computing a GSA of an ontology is that, in current implementations, any reasoner for $\mathcal{L}_S$ must reason over the ontology as a whole. From this observation, the idea for a new notion of approximation, in which we do not reason over the entire ontology but only over portions of it, arises. At the basis of this new notion, which we call *k-approximation*, is the idea of obtaining an approximation of the original ontology by computing the global semantic approximation of each set of k axioms of the original ontology in isolation. Below we give a formal definition of the k-approximation.

In what follows, given an ontology $\mathcal{O}$ and a positive integer k such that $k \leq |\mathcal{O}|$, we denote with $subset_k(\mathcal{O})$ the set of all the sets of cardinality k of axioms of $\mathcal{O}$.

Definition 3. *Let $\mathcal{O}_S$ be a satisfiable $\mathcal{L}_S$-ontology and let $\Sigma_{\mathcal{O}_S}$ be the set of predicate and constant symbols occurring in $\mathcal{O}_S$. Let $\mathcal{U}_k = \{\mathcal{O}_i^j \mid \mathcal{O}_i^j \in globalApx(\mathcal{O}_i, \mathcal{L}_T)$, such that $\mathcal{O}_i \in subset_k(\mathcal{O}_S)\}$. An $\mathcal{L}_T$-ontology $\mathcal{O}_T$ over $\Sigma_{\mathcal{O}_S}$ is a* k-approximation *in $\mathcal{L}_T$ of $\mathcal{O}_S$ if both the following statements hold:*

- *$\bigcap_{\mathcal{O}_i^j \in \mathcal{U}_k} Mod(\mathcal{O}_i^j) \subseteq Mod(\mathcal{O}_T)$;*
- *there is no $\mathcal{L}_T$-ontology $\mathcal{O}'$ over $\Sigma_{\mathcal{O}_S}$ such that $\bigcap_{\mathcal{O}_i^j \in \mathcal{U}_k} Mod(\mathcal{O}_i^j) \subseteq Mod(\mathcal{O}') \subset Mod(\mathcal{O}_T)$.*

The following theorem follows from Theorem 1 and provides a constructive condition for the k-approximation.

Theorem 2. *Let $\mathcal{O}_S$ be a satisfiable $\mathcal{L}_S$-ontology and let $\Sigma_{\mathcal{O}_S}$ be the set of predicate and constant symbols occurring in $\mathcal{O}_S$. An $\mathcal{L}_T$-ontology $\mathcal{O}_T$ over $\Sigma_{\mathcal{O}_S}$ is a k-approximation in $\mathcal{L}_T$ of $\mathcal{O}_S$ if and only if:*

(i) $\mathsf{ES}(\mathcal{O}_T, \mathcal{L}_T) \subseteq \mathsf{ES}(\bigcup_{\mathcal{O}_i \in subset_k(\mathcal{O}_S)} \mathsf{ES}(\mathcal{O}_i, \mathcal{L}_T), \mathcal{L}_T)$;
(ii) there is no $\mathcal{L}_T$-ontology $\mathcal{O}'$ over $\Sigma_{\mathcal{O}_S}$ such that $\mathsf{ES}(\mathcal{O}_T, \mathcal{L}_T) \subset \mathsf{ES}(\mathcal{O}', \mathcal{L}_T) \subseteq \mathsf{ES}(\bigcup_{\mathcal{O}_i \in subset_k(\mathcal{O}_S)} \mathsf{ES}(\mathcal{O}_i, \mathcal{L}_T), \mathcal{L}_T)$.

Proof. $(sketch)$ The proof can be easily adapted from the proof of Theorem 1 by observing that in order to prove the theorem one has to show that:
(a) $\bigcap_{\mathcal{O}_i^j \in \mathcal{U}_k} Mod(\mathcal{O}_i^j) \subseteq Mod(\mathcal{O}_T)$ if and only if $\mathsf{ES}(\mathcal{O}_T, \mathcal{L}_T) \subseteq \mathsf{ES}(\bigcup_{\mathcal{O}_i \in subset_k(\mathcal{O}_S)} \mathsf{ES}(\mathcal{O}_i, \mathcal{L}_T), \mathcal{L}_T)$;
(b) and that there is no $\mathcal{L}_T$-ontology $\mathcal{O}'$ over $\Sigma_{\mathcal{O}_S}$ such that $\bigcap_{\mathcal{O}_i^j \in \mathcal{U}_k} Mod(\mathcal{O}_i^j) \subseteq Mod(\mathcal{O}') \subset Mod(\mathcal{O}_T)$ if and only if there is no $\mathcal{L}_T$-ontology $\mathcal{O}''$ over $\Sigma_{\mathcal{O}_S}$ such that $\mathsf{ES}(\mathcal{O}_T, \mathcal{L}_T) \subset \mathsf{ES}(\mathcal{O}'', \mathcal{L}_T) \subseteq \mathsf{ES}(\bigcup_{\mathcal{O}_i \in subset_k(\mathcal{O}_S)} \mathsf{ES}(\mathcal{O}_i, \mathcal{L}_T), \mathcal{L}_T)$. ■

We note that in the case in which $k = |\mathcal{O}_S|$, the k-approximation actually coincides with the GSA. At the other end of the spectrum, we have the case in which $k = 1$. Here we are treating each axiom α in the original ontology in isolation, i.e., we are considering ontologies formed by a single axiom α. We refer to this approximation as *local semantic approximation* (LSA).

We conclude this section with an example highlighting the difference between the GSA and the LSA.

Example 1. Consider the following OWL 2 ontology $\mathcal{O}$.

$$\begin{array}{lll} \mathcal{O} = \{\, A \sqsubseteq B \sqcup C & B \sqsubseteq D & A \sqsubseteq \exists R.D \\ \quad\; B \sqcap C \sqsubseteq F & C \sqsubseteq D & \exists R.D \sqsubseteq E \,\}. \end{array}$$

The following ontology is a GSA in OWL 2 QL of $\mathcal{O}$.

$$\begin{array}{lllll} \mathcal{O}_{GSA} = \{ & A \sqsubseteq D & B \sqsubseteq D & A \sqsubseteq \exists R & A \sqsubseteq \exists R.D \\ & A \sqsubseteq E & C \sqsubseteq D & D \sqsubseteq F & \}. \end{array}$$

Indeed, it is possible to show that, according to Theorem 1, each axiom entailed by $\mathcal{O}_{GSA}$ is also entailed by $\mathcal{O}$, and that it is impossible to build an OWL 2 QL ontology $\mathcal{O}'$ such that $\mathsf{ES}(\mathcal{O}_{GSA}, OWL\,2\,QL) \subset \mathsf{ES}(\mathcal{O}', OWL\,2\,QL) \subseteq \mathsf{ES}(\mathcal{O}, OWL\,2\,QL)$.

Computing the LSA in OWL 2 QL of $\mathcal{O}$, i.e., its 1-approximation in OWL 2 QL, we obtain the following ontology.

$$\begin{array}{lll} \mathcal{O}_{LSA} = \{ & B \sqsubseteq D & A \sqsubseteq \exists R \\ & C \sqsubseteq D & A \sqsubseteq \exists R.D \,\}. \end{array}$$

It is easy to see that $Mod(\mathcal{O}) \subset Mod(\mathcal{O}_{GSA}) \subset Mod(\mathcal{O}_{LSA})$, which means that the ontology $\mathcal{O}_{GSA}$ approximates $\mathcal{O}$ better than $\mathcal{O}_{LSA}$. This expected result is a consequence of the fact that reasoning over each single axiom in $\mathcal{O}$ in isolation does not allow for the extraction all the OWL 2 QL consequences of $\mathcal{O}$.

Moreover, from Lemma 2, it follows that every $\mathcal{O}' \in globalApx(\mathcal{O}_S, OWL\,2\,QL)$ is logically equivalent to $\mathcal{O}_{GSA}$. □

4 Approximation in OWL 2 QL

In this section we deal with the problem of approximating ontologies in OWL 2 with ontologies in OWL 2 QL.

Based on the characteristics of the OWL 2 QL language, we can give the following theorem.

Algorithm 1: $computeKApx(\mathcal{O}, k)$

```
Input: a satisfiable OWL 2 ontology $\mathcal{O}$, a positive integer $k$ such that $k \leq |\mathcal{O}|$
Output: an OWL 2 QL ontology $\mathcal{O}_{Apx}$
begin
    $\mathcal{O}_{Apx} \leftarrow \emptyset$;
    foreach ontology $\mathcal{O}_i \in subset_k(\mathcal{O}_S)$
        $\mathcal{O}_{Apx} \leftarrow \mathcal{O}_{Apx} \cup \mathsf{ES}(\mathcal{O}_i, OWL\,2\,QL)$;
    return $\mathcal{O}_{Apx}$;
end
```

Theorem 3. *Let $\mathcal{O}_S$ be a satisfiable OWL 2 ontology. Then the OWL 2 QL ontology $\bigcup_{\mathcal{O}_i \in subset_k(\mathcal{O}_S)} \mathsf{ES}(\mathcal{O}_i, OWL\,2\,QL)$ is the k-approximation in OWL 2 QL of $\mathcal{O}_S$.*

Proof. $(sketch)$ To prove the claim, we observe that Lemma 1 holds for OWL 2 QL ontologies, and this guarantees that for every OWL 2 ontology $\mathcal{O}_S$, there exists at least one OWL 2 QL ontology which is its GSA, i.e., $globalApx(\mathcal{O}_S, OWL\ 2\ QL) \neq \emptyset$. Moreover, we have that since OWL 2 QL is closed, for Lemma 2, all ontologies in $\mathsf{ES}(\mathcal{O}_S, OWL\ 2\ QL)$ are pairwise logically equivalent. Another consequence of the fact that OWL 2 QL is closed is that, whichever language the original ontology $\mathcal{O}_S$ is expressed in, $\mathsf{ES}(\mathcal{O}_S, OWL\,2\,QL)$ is an OWL 2 QL ontology. Furthermore, given a set of OWL 2 QL ontologies, the union of these ontologies is still an OWL 2 QL ontology. From these observations, it is easy to see that, given an OWL 2 ontology $\mathcal{O}_S$ and an integer $k \leq |\mathcal{O}_S|$, the set $\bigcup_{\mathcal{O}_i \in subset_k(\mathcal{O}_S)} \mathsf{ES}(\mathcal{O}_i, OWL\,2\,QL)$ satisfies conditions (i) and (ii) of Theorem 2. Hence, we have the claim. ■

Notably, we observe that for $k = |\mathcal{O}_S|$ the k-approximation $\mathcal{O}_T$ in OWL 2 QL of $\mathcal{O}_S$ is unique and coincides with its entailment set in OWL 2 QL. This means that $\mathcal{O}_T$ is also the approximation in OWL 2 QL of $\mathcal{O}_S$ according to the notion of approximation presented in [14]. Therefore, all the properties that hold for the semantics in [14] also hold for the GSA. In particular, the evaluation of a conjunctive query q without non-distinguished variables over $\mathcal{O}_S$ coincides with the evaluation of q over $\mathcal{O}_T$ (Theorem 5 in [14]).

From Theorem 3, one can easily come up with Algorithm 1 for computing the k-approximation of an $\mathcal{L}_S$-ontology $\mathcal{O}_S$ in OWL 2 QL. The algorithm first computes every subset with size k of the original ontology $\mathcal{O}_S$. Then, it computes the ontology which is the result of the k-approximation in OWL 2 QL of the ontology in input as the union of the entailment sets with respect to OWL 2 QL of each such subset. A naive algorithm for computing the entailment set with respect to OWL 2 QL can be easily obtained from the one given in [14] for *DL-Lite* languages. We can summarize it as follows. Let $\mathcal{O}$ be an ontology and let $\Sigma_{\mathcal{O}}$ be the set of predicate and constant symbols occurring in $\mathcal{O}$. The algorithm first computes the set Γ of axioms in $\Phi_{OWL\ 2\ QL}$ which can be built over $\Sigma_{\mathcal{O}}$, and then, for each axiom $\alpha \in \Gamma$ such that $\mathcal{O} \models \alpha$, adds α to the set $\mathsf{ES}(\mathcal{O}, OWL\,2\,QL)$. In practice, to check if $\mathcal{O} \models \alpha$ one can use an OWL 2 reasoner.

Since each invocation of the OWL 2 reasoner is N2EXPTIME, the computation of the entailment set can be very costly [4].

A more efficient technique for its computation is given in [8], where the idea is to limit the number of invocations to the OWL 2 reasoner by exploiting the knowledge acquired through a preliminary exploration of the ontology. To understand the basic idea behind this technique, consider, for example, an ontology $\mathcal{O}$ that entails the inclusions $A_1 \sqsubseteq A_2$ and $P_1 \sqsubseteq P_2$, where A_1 and A_2 are concepts and P_1 and P_2 are roles. Exploiting these inclusions we can deduce the hierarchical structure of the general concepts that can be built on these four predicates. For instance, we know that $\exists P_2.A_2 \sqsubseteq \exists P_2$, that $\exists P_2.A_1 \sqsubseteq \exists P_2.A_2$, that $\exists P_1.A_1 \sqsubseteq \exists P_2.A_1$, and so on. To obtain the entailed inclusion axioms, we begin by invoking the OWL 2 reasoner, asking for the children of the general concepts which are in the highest position in the hierarchy. So we first compute the subsumees of $\exists P_2$ through the OWL 2 reasoner. If there are none, we avoid invoking the reasoner asking for the subsumees of $\exists P_2.A_2$ and so on. Regarding the entailed disjointness axioms, we follow the same approach but starting from the lowest positions in the hierarchy.

The following theorem establishes correctness and termination of algorithm $computeKApx$.

Theorem 4. *Let $\mathcal{O}_S$ be a satisfiable OWL 2 ontology. $computeKApx(\mathcal{O}_S, k)$ terminates and computes the k-approximation in OWL 2 QL of $\mathcal{O}_S$.*

Proof. $(sketch)$ Termination of $computeKApx(\mathcal{O}_S, k)$ directly follows from the fact that $\mathcal{O}_S$ is a finite set of axioms and that, for each $\mathcal{O}_i \in subset_k(\mathcal{O}_S)$, $\mathsf{ES}(\mathcal{O}_i, OWL\ 2\ QL)$ can be computed in finite time. The correctness of the algorithm directly follows from Theorem 3.

5 Experiments

In this section we present the experimental tests that we have performed for the approximation of a suite of OWL 2 ontologies into OWL 2 QL through the two notions of approximation we have introduced earlier.

We notice that by choosing a value for k different from $|\mathcal{O}_S|$, the computation of the entailment set becomes easier. However, observing Algorithm 1, the number of times that this step must be repeated can grow very quickly. In fact, the number of sets of axioms in $subset_k(\mathcal{O}_S)$ is equal to the binomial coefficient of $|\mathcal{O}_S|$ over k, and therefore for large ontologies this number can easily become enormous, and this can be in practice a critical obstacle in the computation of the k-approximation.

For this reason, in these experiments we have focused on comparing the GSA (k-approximation with $k = |\mathcal{O}_S|$) to the LSA (k-approximation with $k = 1$), and we reserve the study of efficient techniques for k-approximation with $1 < k < |\mathcal{O}_S|$ for future works. Furthermore, to provide a standard baseline against which to compare the results of the GSA and LSA, we have compared both our approaches with a syntactic sound approximation approach, consisting in first normalizing the axioms in the ontology and then eliminating the ones that are not syntactically compliant with OWL 2 QL. We will refer to this approach as "SYNT".

Ontology	Expressiveness	Axioms	Concepts	Roles	Attributes	OWL2 QL Axioms
Homology	$\mathcal{ALC}$	83	66	0	0	83
Cabro	$\mathcal{ALCHIQ}$	100	59	13	0	99
Basic vertebrate anatomy	$\mathcal{SHIF}$	108	43	14	0	101
Fungal anatomy	$\mathcal{ALEI}+$	115	90	5	0	113
Pmr	$\mathcal{ALU}$	163	137	16	0	159
Ma	$\mathcal{ALE}+$	168	166	8	0	167
General formal Ontology	$\mathcal{SHIQ}$	212	45	41	0	167
Cog analysis	$\mathcal{SHIF}(\mathcal{D})$	224	92	37	9	213
Time event	$\mathcal{ALCROIQ}(\mathcal{D})$	229	104	28	7	170
Spatial	$\mathcal{ALEHI}+$	246	136	49	0	155
Translational medicine	$\mathcal{ALCRIF}(\mathcal{D})$	314	225	18	6	298
Biopax	$\mathcal{SHIN}(\mathcal{D})$	391	69	55	41	240
Vertebrate skeletal anatomy	$\mathcal{ALER}+$	455	314	26	0	434
Image	$\mathcal{S}$	548	624	2	0	524
Protein	$\mathcal{ALCF}(\mathcal{D})$	691	45	50	133	490
Pizza	$\mathcal{SHOIN}$	712	100	8	0	660
Ontology of physics for biology	$\mathcal{ALCHIQ}(\mathcal{D})$	954	679	33	3	847
Plant trait	$\mathcal{ALE}+$	1463	1317	4	0	1461
Dolce	$\mathcal{SHOIN}(\mathcal{D})$	1667	209	313	4	1445
Ont. of athletic events	$\mathcal{ALEH}$	1737	1441	15	1	1722
Neomark	$\mathcal{ALCHQ}(\mathcal{D})$	1755	55	105	488	842
Pato	$\mathcal{SH}$	1979	2378	36	0	1779
Protein Modification	$\mathcal{ALE}+$	1986	1338	8	0	1982
Po anatomy	$\mathcal{ALE}+$	2128	1294	11	0	2064
Lipid	$\mathcal{ALCHIN}$	2375	716	46	0	2076
Plant	$\mathcal{S}$	2615	1644	16	0	2534
Mosquito anatomy	$\mathcal{ALE}+$	2733	1864	5	0	2732
Idomal namespace	$\mathcal{ALER}+$	3467	2597	24	0	3462
Cognitive atlas	$\mathcal{ALC}$	4100	1701	6	0	3999
Genomic	$\mathcal{ALCQ}$	4322	2265	2	0	3224
Mosquito insecticide resistance	$\mathcal{ALE}+$	4413	4362	21	0	4409
Galen-A	$\mathcal{ALEHIF}+$	4979	2748	413	0	3506
Ni gene	$\mathcal{SH}$	8835	4835	8	0	8834
Fyp	$\mathcal{SH}$	15105	4751	69	0	12924
Fly anatomy	$\mathcal{SH}$	20356	8064	72	0	20353
EL-Galen	$\mathcal{ALEH}+$	36547	23136	950	0	25138
Galen full	$\mathcal{ALEHIF}+$	37696	23141	950	0	25613
Pr reasoned	$\mathcal{S}$	46929	35470	14	0	40031
Snomed fragment for FMA	$\mathcal{ALER}$	73543	52635	52	0	35004
Gene	$\mathcal{SH}$	73942	39160	12	0	73940
FMA OBO	$\mathcal{ALE}+$	119560	75139	2	0	119558

Table 1: Characteristics of the ontologies used in the GSA and LSA tests.

The suite of ontologies used during testing contains 41 ontologies and was assembled from the Bioportal ontology repository[2]. The ontologies that compose this suite were selected to test the scalability of our approaches both to larger ontologies and to ontologies formulated in more expressive languages. In Table 1 we present the most relevant metrics of these ontologies.

All tests were performed on a DELL Latitude E6320 notebook with Intel Core i7-2640M 2.8Ghz CPU and 4GB of RAM, running Microsoft Windows 7 Premium operating system, and Java 1.6 with 2GB of heap space. Timeout was set at eight hours, and execution was aborted if the maximum available memory was exhausted. The tool used in the experiments and the suite of ontologies are available at `http://diag.uniroma1.it/~mora/ontology_approximation/iswc2014/`.

As mentioned in Section 4, the computation of the entailment set involves the use of an external OWL 2 reasoner. Therefore, the performance and the results of the computed approximations are greatly effected by the choice of the reasoner. For our tests, we have used the Pellet[3] OWL 2 reasoner (version v.2.3.0.6).

In Table 2 we present the results of the evaluation. An analysis of these results leads to the following observations.

Firstly, we were able to compute the GSA for 26 out of the 41 tested ontologies. For the remaining fifteen, this was not possible, either due to the size of the ontology, in terms of the number of its axioms, e.g., the FMA 2.0 or Gene ontologies, which have more than seventy thousand and one hundred thousand axioms, respectively, or due to its high expressivity, e.g., the Dolce ontology or the General formal ontology. The LSA approach is instead always feasible, it is quicker than the GSA approach for all but one of the tested ontologies, and it is overall very fast: no ontology took more than 250 seconds to approximate with the LSA.

Secondly, it is interesting to observe the comparison between the quality of the approximation that one can obtain through the LSA with respect to that obtained through the GSA. This relationship answers the question of whether the ontology obtained through the LSA (the "LSA ontology") is able to capture a significant portion of the one obtained through the GSA (the "GSA ontology"). Our tests in fact confirm that this is the case: out of the 26 ontologies for which we were able to compute the GSA, in only four cases the LSA ontology entails less than 60 percent of the axioms of the GSA ontology, while in twenty cases it entails more than 90 percent of them. The average percentage of axioms in the original ontologies entailed by the GSA ontologies is roughly 80 percent, and of the axioms of the GSA ontologies entailed by the LSA ontologies is roughly 87 percent.

Furthermore, the LSA provides a good approximation even for those ontologies for which the GSA is not computable. In fact, Table 3 shows the percentage of axioms of the original ontology that are entailed by the LSA ontology. Out of the twelve ontologies for which we were able to obtain this value (the remaining three ontologies caused an "out of memory" error), only in three cases it was less than 60 percent, while in four cases it was higher than 80 percent. These results are particularly interesting with respect to those ontologies for which the GSA approach is not feasible due to their

[2] `http://bioportal.bioontology.org/`
[3] `http://clarkparsia.com/pellet/`

Ontology	GSA axioms	GSA entails original (%)	LSA axioms	LSA entails GSA (%)	SYNT axioms	SYNT entails GSA (%)	SYNT entails LSA (%)	GSA time (s)	LSA time (s)
Homology	83	100	83	100	83	100	100	1	4
Cabro	233	96	121	100	100	100	100	4	2
Basic vertebrate anatomy	192	93	141	97	71	56	67	3	3
Fungal anatomy	318	98	140	69	113	69	100	2	2
Pmr	162	97	159	98	159	98	100	2	2
Ma	411	99	240	95	167	96	100	4	4
General formal ontology	–	–	286	–	177	–	100	–	6
Cog analysis	104407	75	474	46	215	1	82	36	7
Time event	93769	71	662	99	196	1	58	45	11
Spatial	510	63	371	86	155	42	52	9	4
Translational medicine	4089	86	505	99	275	30	64	19	7
Biopax	2182057	–	3217	–	251	–	81	–	11
Vertebrate skeletal anatomy	9488	95	581	92	434	57	99	27	5
Image	1016	95	596	98	571	98	100	178	5
Protein	–	–	10789	–	475	–	88	–	20
Pizza	2587	91	755	92	678	92	99	7	4
Ont. of physics for biology	1789821	–	1505	–	1241	–	100	–	7
Plant trait	2370	99	1496	99	1461	100	100	10	9
Dolce	–	–	2959	–	1555	–	100	–	8
Ontology of athletic events	5073	99	2392	99	1731	92	100	42	9
Neomark	–	–	39807	–	1723	–	63	–	50
Pato	4066	89	2209	100	1976	78	99	99	18
Protein Modification	2195	99	2001	100	1982	100	100	12	19
Po anatomy	11486	96	2783	77	2078	78	100	455	18
Lipid	14659	87	3165	97	2759	89	97	47	10
Plant	18375	96	3512	80	2574	81	100	929	15
Mosquito anatomy	21303	99	4277	43	2732	44	100	214	16
Idomal namespace	67417	99	4259	98	3461	59	100	496	16
Cognitive atlas	7449	97	5324	100	1364	26	30	162	17
Genomic	–	–	86735	–	85037	–	98	–	54
Mosquito insecticide res.	6794	99	4502	100	4409	100	100	86	14
Galen-A	–	–	8568	–	4971	–	90	–	26
Ni gene	46148	99	10415	90	8834	91	100	472	32
Fyp	–	–	19675	–	11800	–	82	–	43
Fly anatomy	460849	99	28436	67	20346	67	100	25499	45
EL-Galen	–	–	70272	–	43804	–	89	–	59
Galen full	–	–	72172	–	44279	–	89	–	61
Pr reasoned	–	–	56085	–	47662	–	100	–	93
Snomed fragment for FMA	–	–	140629	–	101860	–	76	–	250
Gene	–	–	86292	–	73940	–	100	–	178
FMA OBO	–	–	143306	–	119558	–	100	–	113

Table 2: Results of the GSA, LSA, and SYNT. The values represent, from left to right, the number of axioms in the ontology obtained through the GSA, the percentage of axioms of the original ontology that are entailed by the ontology obtained through the GSA, the number of axioms in the ontology obtained through the LSA, the percentage of axioms of the ontology obtained through the GSA that are entailed by the LSA, the number of axioms in the ontology obtained by the SYNT, the percentage of axioms of the ontology obtained through the GSA that are entailed by the ontology obtained through the SYNT, the percentage of axioms of the ontology obtained through the LSA that are entailed by the ontology obtained through the SYNT, and finally the GSA time and the LSA time (both in seconds).

Ontology	Original axioms	LSA axioms	LSA entails original (%)	LSA time (s)
General formal ontology	212	264	67	6
Biopax	391	3204	53	11
Protein	691	10720	47	20
Ontology of physics for biology	954	1074	75	7
Dolce	1667	2914	78	8
Neomark	1755	38966	46	50
Genomic	4322	9844	65	54
Galen-A	4979	8568	70	26
Fyp	15105	19672	85	43
EL-Galen	36547	70272	–	59
Galen full	37696	72172	–	61
Pr reasoned	46929	55391	83	93
SNOMED fragment for FMA	73543	140629	–	250
Gene	73942	86289	99	178
FMA OBO	119560	143306	99	113

Table 3: LSA results for ontologies for which the GSA is not computable.

complexity, as is the case for example for the Dolce ontology, for Galen-A, and for the Ontology of physics for biology. Indeed, even though these ontologies are expressed in highly expressive DL languages, the structure of the axioms that compose them is such that reasoning on each of them in isolation does not lead to much worse approximation results than reasoning on the ontology as a whole: for the nine smallest ontologies in Table 3, for which the GSA fails not because of the size of the ontology, the average percentage is 68.6.

Finally, both the GSA and LSA compare favorably against the syntactic sound approximation approach. In fact, the average percentage of axioms in the LSA and GSA ontologies that are entailed by the ontologies obtained through the SYNT approach are respectively roughly 90 percent and 72 percent. While the latter result is to be expected, the former is quite significant, even more so when one considers that the LSA is very fast. Indeed, a "gain" of 10 percent of axiom entailments by the LSA with respect to the SYNT in the case of large ontologies such as Galen and Snomed translates to tens of thousands of preserved axioms in very little computation time.

In conclusion, the results gathered from these tests corroborate the usefulness of both the global semantic approximation and the local semantic approximation approaches. The former provides a maximal sound approximation in the target language of the original approach, and is in practice computable in a reasonable amount of time for the majority of the tested ontologies. The latter instead represents a less optimal but still very effective solution for those ontologies for which the GSA approach goes beyond the capabilities of the currently-available ontology reasoners.

6 Conclusions

In this paper we have addressed the problem of ontology approximation in Description Logics and OWL, presenting (i) a parameterized semantics for computing sound approximations of ontologies, (ii) algorithms for the computation of approximations (the GSA and the LSA) of OWL 2 ontologies in OWL 2 QL, and (iii) an extensive experimental evaluation of the above techniques, which empirically proves the validity of our approach.

The present work can be extended in several ways. First, while we have focused on sound approximations, it would be interesting to also consider complete approximations of ontologies. Also, we would like to study the development of techniques for k-approximations different from GSA and LSA, i.e., for k such that $1 < k < |\mathcal{O}_S|$, as well as to analyze the possibility of integrating ontology module extraction techniques in our approach. Then, this work has not addressed the case when there are differences in the semantic assumptions between the source and the target ontology languages. For instance, differently from OWL 2 and its profiles, some DLs (e.g., *DL-Lite*$_A$ [15]) adopt the Unique Name Assumption (UNA). This makes our approach not directly applicable, for instance, if we consider OWL 2 as the source language and *DL-Lite*$_A$ as the target language. The reason is that the UNA implies some axioms (inequalities between individuals) that can be expressed in OWL 2 but cannot be expressed in *DL-Lite*$_A$. We aim at extending our approach to deal with the presence of such semantic discrepancies in the ontology languages. Finally, we are very interested in generalizing our approach to a full-fledged ontology-based data access scenario [15], in which data sources are connected through declarative mappings to the ontology. In that context, it might be interesting to use both the ontology and the mappings in the target OBDA specification to approximate a given ontology in the source OBDA specification.

Acknowledgments. This research has been partially supported by the EU under FP7 project Optique (grant n. FP7-318338).

References

1. Natalia Antonioli, Francesco Castanò, Cristina Civili, Spartaco Coletta, Stefano Grossi, Domenico Lembo, Maurizio Lenzerini, Antonella Poggi, Domenico Fabio Savo, and Emanuela Virardi. Ontology-based data access: the experience at the Italian Department of Treasury. In *Proc. of the Industrial Track of CAiSE 2013*, volume 1017 of *CEUR*, `ceur-ws.org`, pages 9–16, 2013.
2. Alessandro Artale, Diego Calvanese, Roman Kontchakov, and Michael Zakharyaschev. The *DL-Lite* family and relations. *J. of Artificial Intelligence Research*, 36:1–69, 2009.
3. Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2003.
4. Elena Botoeva, Diego Calvanese, and Mariano Rodriguez-Muro. Expressive approximations in DL-Lite ontologies. *Proc. of AIMSA 2010*, pages 21–31, 2010.
5. Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *J. of Automated Reasoning*, 39(3):385–429, 2007.

6. Diego Calvanese, Martin Giese, Peter Haase, Ian Horrocks, Thomas Hubauer, Yannis E. Ioannidis, Ernesto Jiménez-Ruiz, Evgeny Kharlamov, Herald Kllapi, Johan W. Klüwer, Manolis Koubarakis, Steffen Lamparter, Ralf Möller, Christian Neuenstadt, T. Nordtveit, Özgür L. Özçep, Mariano Rodriguez-Muro, Mikhail Roshchin, Domenico Fabio Savo, Michael Schmidt, Ahmet Soylu, Arild Waaler, and Dmitriy Zheleznyakov. Optique: OBDA Solution for Big Data. In *ESWC (Satellite Events)*, volume 7955 of *Lecture Notes in Computer Science*, pages 293–295. Springer, 2013.
7. Cristina Civili, Marco Console, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Lorenzo Lepore, Riccardo Mancini, Antonella Poggi, Riccardo Rosati, Marco Ruzzi, Valerio Santarelli, and Domenico Fabio Savo. MASTRO STUDIO: Managing ontology-based data access applications. *PVLDB*, 6:1314–1317, 2013.
8. Marco Console, Valerio Santarelli, and Domenico Fabio Savo. Efficient approximation in DL-Lite of OWL 2 ontologies. In *Proc. of DL 2013*, volume 1014 of *CEUR Workshop Proceedings*, pages 132–143, 2013.
9. Pascal Hitzler, Markus Krötzsch, Bijan Parsia, Peter F. Patel-Schneider, and Sebastian Rudolph. *OWL 2 Web Ontology Language: Primer*. W3C Recommendation, 2012. Available at http://www.w3.org/TR/2012/REC-owl2-primer-20121211/.
10. Ian Horrocks, Oliver Kutz, and Ulrike Sattler. The even more irresistible $\mathcal{SROIQ}$. In *Proc. of KR 2006*, pages 57–67, 2006.
11. Carsten Lutz, Inanç Seylan, and Frank Wolter. An Automata-Theoretic Approach to Uniform Interpolation and Approximation in the Description Logic EL. In *Proc. of KR 2012*. AAAI Press, 2012.
12. Boris Motik, Bernardo Cuenca Grau, Ian Horrocks, Zhe Wu, Achille Fokoue, and Carsten Lutz. *Owl 2 web ontology language: Profiles (2nd edition)*. W3C Recommendation, 2012. Available at http://www.w3.org/TR/owl2-profiles/.
13. Boris Motik, Bijan Parsia, and Peter F. Patel-Schneider. *OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax*. W3C Recommendation, 2012. Available at http://www.w3.org/TR/2012/REC-owl2-syntax-20121211/.
14. Jeff Z. Pan and Edward Thomas. Approximating OWL-DL ontologies. In *Proc. of AAAI 2007*, pages 1434–1439, 2007.
15. Antonella Poggi, Domenico Lembo, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. Linking data to ontologies. *J. on Data Semantics*, X:133–173, 2008.
16. Mariano Rodriguez-Muro, Roman Kontchakov, and Michael Zakharyaschev. Ontology-based data access: Ontop of databases. In *Proc. of ISWC 2013*, volume 8218 of *LNCS*, pages 558–573. Springer, 2013.
17. Tuvshintur Tserendorj, Sebastian Rudolph, Markus Krötzsch, and Pascal Hitzler. Approximate OWL-reasoning with Screech. In *Proc. of RR 2008*, pages 165–180. Springer, 2008.
18. Holger Wache, Perry Groot, and Heiner Stuckenschmidt. Scalable instance retrieval for the semantic web by approximation. In *Proc. of WISE 2005*, pages 245–254. Springer, 2005.

Appendix I

DL 2013: Ontology Approximation

This appendix reports the paper:

– Marco Console, Valerio Santarelli, Domenico Fabio Savo. Efficient Approximation in DL-Lite of OWL 2 Ontologies. Description Logics 2013.

Efficient approximation in *DL-Lite* of OWL 2 ontologies

Marco Console, Valerio Santarelli, Domenico Fabio Savo

Dipartimento di Ingegneria Informatica Automatica e Gestionale "Antonio Ruberti"
SAPIENZA Università di Roma
lastname@dis.uniroma1.it

Abstract. Ontologies, as a conceptualization of a domain of interest, can be used for different objectives, such as for providing a formal description of the domain of interest for documentation purposes, or for providing a mechanism for reasoning upon the domain. For instance, they are the core element of the Ontology-Based Data Access paradigm, in which the ontology is utilized as a conceptual view, allowing user access to the underlying data sources. With the aim to use an ontology as a formal description of the domain of interest, the use of expressive languages proves to be useful. If instead the goal is to use the ontology for reasoning tasks which require low computational complexity, the high expressivity of the language used to model the ontology may be of hindrance. In this scenario, the approximation of ontologies expressed in very expressive languages through ontologies expressed in languages which keep the computational complexity of the reasoning tasks low is pivotal. In this work we present our notion of ontology approximation and present an algorithm for computing the approximation of OWL 2 ontologies by means of DL-Lite TBoxes. Moreover, we provide optimization techniques for this computation, and discuss the results of the implementation of these techniques.

1 Introduction

Ontologies, as a shared conceptualization of a domain of interest, are commonly recognized as powerful tools in support of the information systems of organizations [6,4,10]. On one hand, they can be used for providing a formal description of the domain of interest, and thus represent valuable documentation for an organization. On the other hand, they provide a mechanism for reasoning upon the domain with different objectives. For instance, they are the core element of the Ontology-Based Data Access (OBDA) [13,4] paradigm, in which the ontology is utilized as a conceptual view of the underlying data sources, granting the user the ability to retrieve information without specific knowledge on how this information is organized and where it is stored.

With the aim to use an ontology as a formal description of the domain of interest, the use of expressive languages, such as the OWL 2 Web Ontology Language [9], proves to be useful. The expressivity of such languages allows the ontology designer to obtain a precise formalization of the domain. If instead the goal is to use the ontology for reasoning tasks, the high expressivity of the language used to model the ontology may be of hindrance. In particular, when wishing to access large quantities of data through the ontology, as in OBDA, the computational cost of very expressive languages such as OWL

2 is prohibitive. In these cases, it is necessary to recur to less expressive languages, thus resigning to having less complete representations of the domain of interest.

One possible solution to this issue is to allow the ontology designer the use of expressive languages to define ontologies that model the domain in great detail for the purpose of documentation and of other tasks that do not require strong computational effort, while adopting, for all those reasoning tasks in which particular computational properties are required, such as OBDA, descriptions of the domain of interest obtained through less expressive languages.

Following this approach, the notion of approximation becomes pivotal. The goal of approximation is, given a ontology $\mathcal{O}$ in a language $\mathcal{L}$, to compute an ontology $\mathcal{O}'$ in a target language $\mathcal{L}'$, in which as much as possible of the semantics of $\mathcal{O}$ is preserved. In general, various approaches can be adopted towards this goal [17,14,12,18,3]. Among these, the approaches in which we are most interested are those which aim to obtain the so-called *semantic* approximation [14,3,12]. Here, as opposed to those approaches which aim at a syntactic approximation [17,18], the computation of the ontology which is the approximation of the original one is obtained through the semantics of this original ontology, and not only through its syntactic representation.

In this paper, we focus on the semantic approximation of an ontology for OBDA applications. Thus, we study approaches for approximating ontologies in very expressive languages with ontologies in languages that, characterized by low reasoning complexity, are suitable for query answering purposes. The most significant works in which this problem is studied are [12] and [3], in which the proposed approaches can be traced back to the work of Selman and Kautz [14].

These approaches, as is ours, are based on the notion of *entailment set*. Given an ontology $\mathcal{O}$ in a language $\mathcal{L}$, the entailment set of $\mathcal{O}$ in a target language $\mathcal{L}'$ is the set of all the assertions expressible in $\mathcal{L}'$ over Σ that are entailed by $\mathcal{O}$. The idea behind our approach is that an approximation in $\mathcal{L}'$ of $\mathcal{O}$ is an ontology $\mathcal{O}'$ whose entailment set minimally differs from the entailment set of $\mathcal{O}$ in $\mathcal{L}'$.

Since OWL 2 is the W3C standard language for expressing ontologies, it is often used as the expressive language for formulating ontologies describing the domain of interest. On the other hand, in the context of OBDA, one naturally focuses on the logics of the *DL-Lite* family [5]. This is a family of DLs specifically designed to keep all reasoning tasks polynomially tractable in the size of the data, and is thus suitable for OBDA. For this reason, in this work we study the problem of approximating OWL 2 ontologies with ontologies in *DL-Lite*. To this aim we provide an algorithm for the computation of these approximations, and an optimized technique for the computation of the entailment set of an OWL 2 ontology in *DL-Lite*, which can be used efficiently in practice.

The rest of the paper is organized as follows. In Section 2, we provide some useful notions for the paper. In Section 3 we study the problem of ontology approximation, and introduce our notion of approximation. In Section 4 we focus on the approximation in *DL-Lite* of OWL 2 ontologies. In Section 5 we describe our technique for optimizing the computation of the entailment set of an OWL 2 ontology in *DL-Lite*, and present the results of our experimentation. Finally, in Section 6 we conclude the paper.

2 Preliminaries

Description Logics (DLs) [2] are logics that allow one to represent the domain of interests in terms of *concepts*, denoting sets of objects, *value-domains*, denoting sets of values, *attributes*, denoting binary relations between objects and values, and *roles* denoting binary relations over objects.

Let Σ be a signature of symbols for individual (object and value) constants and atomic elements, i.e., concepts, value-domains, attributes, and roles. If $\mathcal{L}$ is a DL, then an ontology $\mathcal{O}$ in $\mathcal{L}$ (or $\mathcal{L}$ ontology) over Σ is the set $\mathcal{T} \cup \mathcal{A}$, where $\mathcal{T}$, the *TBox*, is a finite set of intensional assertions over Σ expressed in $\mathcal{L}$, and $\mathcal{A}$, the *ABox*, is a finite set of instance assertions, i.e, assertions on individuals, over Σ. Different DLs allow for different kinds of TBox and/or ABox assertions and allow for different manners in which these can be combined for obtaining TBoxes and ABoxes in the specific DL.

The semantics of a DL ontology is given in terms of first-order (FOL) interpretations (cf. [2]). We denote with $Mod(\mathcal{O})$ the set of models of $\mathcal{O}$, i.e., the set of FOL interpretations that satisfy all the assertions in $\mathcal{T}$ and $\mathcal{A}$, where the definition of satisfaction depends on the kind of expressions and assertions in the specific DL language in which $\mathcal{O}$ is specified. As usual, a ontology $\mathcal{O}$ is said to be *satisfiable* if it admits at least one model, and $\mathcal{O}$ is said to *entail* a First-Order Logic (FOL) sentence ϕ, denoted $\mathcal{O} \models \phi$, if $\phi^{\mathcal{I}} = \mathit{true}$ for all $\mathcal{I} \in Mod(\mathcal{O})$. Moreover, given to ontology $\mathcal{O}$ and $\mathcal{O}'$, we say that $\mathcal{O}$ and $\mathcal{O}'$ are logically equivalent if $Mod(\mathcal{O}) = Mod(\mathcal{O}')$.

In this work we mainly focus on two specific languages, which are OWL 2, the official ontology language of the World Wide Web Consortium (W3C) [9], and *DL-Lite*$_A$, a member of the *DL-Lite* family [5], which is a family of tractable DLs particularly suited for dealing with ontologies with very large ABoxes, and is at the basis of OWL 2 QL, one of the profiles of OWL 2.

The Web Ontology Language (version 2), or simply OWL 2, is an ontology language for the Semantic Web with formally defined meaning. OWL 2 provides for describing the domain of interest in terms of concepts, roles, attributes, individuals, and data values [9]. Due to the limitation of space, here we do not provide a complete description of OWL 2, but we refer the reader to the official W3C specification [11].

We now present the syntax of the DL *DL-Lite*$_A$. As for the semantics, we apply the general definitions given above, and refer the reader to [13] for the precise description of the semantics of a *DL-Lite*$_A$ ontology.

In what follows we adopt the following notation: A, P, and U are symbols in Σ denoting respectively an atomic concept, an atomic role and an atomic attribute; $T_1, \ldots, T_n$ are n pairwise disjoint unbounded value-domains; B denotes a *basic concept*; C a *general concept*; Q a *basic role*; R a *general role*; V a *general attribute*; E a *basic value-domain*; and F a *value-domain expression*.

Expressions in *DL-Lite*$_A$ are formed according to the following syntax:

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

where: P^- denotes the inverse of P, $\exists Q$, or *unqualified existential restriction* denotes the objects related to some object by the role Q, $\neg$ denotes negation, $\delta(U)$ denotes the *domain* of U, i.e., the set of objects that U relates to values, and $\rho(U)$ denotes the *range* of U, i.e., the set of values related to objects by U. The concept $\exists Q.C$, or *qualified existential restriction*, denotes the *qualified domain* of Q with respect to C, i.e., the set of objects that Q relates to some instance of C. Similarly, $\delta_F(U)$ denotes the *qualified domain* of U with respect to a value-domain F, i.e., the set of objects that U relates to some value in F. We refer to the qualified existential restriction expression $\exists Q_1.\ldots.\exists Q_n.C$, where C is not a qualified existential restriction, as an existential role chain of depth n.

A *DL-Lite$_A$* TBox assertion is an assertion of the form:

$$B \sqsubseteq C \qquad Q \sqsubseteq R \qquad E \sqsubseteq F \qquad U \sqsubseteq V \qquad (\mathsf{funct}\ Q) \qquad (\mathsf{funct}\ U)$$

From left to right, the first four assertions denote inclusions between concepts, roles, value-domains, and attributes, respectively. The last two assertions denote functionality on roles and on attributes.

A *DL-Lite$_A$* TBox is a finite set of assertions of the form above, where suitable limitations in the combination of such assertion are imposed. Given a TBox $\mathcal{T}$ and a role P (resp. an attribute U), we say that P (resp. U) is *primitive* in $\mathcal{T}$ if P does not appear in $\mathcal{T}$ positively in the right-hand side of any role (resp. attribute) inclusion assertion and in any qualified existential restriction. In a *DL-Lite$_A$* TBox, a role P (resp, attribute U) that is not primitive in $\mathcal{T}$ cannot appear either directly nr inversely in a functionality assertion.

A *DL-Lite$_A$* ABox $\mathcal{A}$ is a finite set of assertions of the form $A(a)$, $P(a, b)$, and $U(a, v)$, where A, P, and U are as above, a and b are object constants while v is a value constant.

3 Approximation of DL ontologies

In this section, we present our notion of approximation of an ontology expressed in a language $\mathcal{L}$ in a target language $\mathcal{L}'$, and then we provide a comparison between our notion and others proposed in literature.

Ontology approximation. In what follows, $\mathcal{O}$ is a satisfiable $\mathcal{L}$ ontology, and $\mathcal{L}'$ a language not necessarily different from $\mathcal{L}$.

First of all, we need to introduce the notion of *entailment set* [12] of a satisfiable ontology with respect to a language.

Definition 1. *Let $\mathcal{O}$ be a satisfiable ontology expressed in a language $\mathcal{L}$ over a signature Σ, and let $\mathcal{L}$' be a language, not necessarily different from $\mathcal{L}$. The* entailment set *of $\mathcal{O}$ with respect to $\mathcal{L}'$, denoted as $\mathsf{ES}(\mathcal{O}, \mathcal{L}')$, is the set which contains all $\mathcal{L}'$ axioms over Σ that are entailed by $\mathcal{O}$.*

In other words, we say that an axiom α belongs to the entailment set of an ontology $\mathcal{O}$ with respect to a language $\mathcal{L}'$, if α is an $\mathcal{L}'$ axiom built over the alphabet of $\mathcal{O}$ and for each interpretation $\mathcal{I} \in Mod(\mathcal{O})$ we have that $\mathcal{I} \models \alpha$. Clearly, given an ontology $\mathcal{O}$ and a language $\mathcal{L}'$, the entailment set of $\mathcal{O}$ with respect to $\mathcal{L}'$ is unique.

With the notion of entailment set in place, we can define the notion of approximation in $\mathcal{L}'$ of $\mathcal{O}'$.

Definition 2. *Let $\mathcal{O}$ be an ontology over a signature Σ. A satisfiable $\mathcal{L}'$ ontology $\mathcal{O}'$ over Σ is an approximation in $\mathcal{L}'$ of $\mathcal{O}$ if both the following statements hold:*

(i) $\mathsf{ES}(\mathcal{O}', \mathcal{L}') \subseteq \mathsf{ES}(\mathcal{O}, \mathcal{L}')$;
(ii) *there is no satisfiable $\mathcal{L}'$ ontology $\mathcal{O}''$ such that $\mathsf{ES}(\mathcal{O}', \mathcal{L}') \subset \mathsf{ES}(\mathcal{O}'', \mathcal{L}') \subseteq \mathsf{ES}(\mathcal{O}, \mathcal{L}')$.*

In other words, the above definition states that a satisfiable ontology $\mathcal{O}'$ is an approximation in $\mathcal{L}'$ of $\mathcal{O}$, if $\mathcal{O}'$ is an ontology in $\mathcal{L}'$, and there is no a satisfiable ontology $\mathcal{O}''$ in $\mathcal{L}'$ whose entailment set in $\mathcal{L}'$ is "nearer" to the entailment set of $\mathcal{O}$ in $\mathcal{L}'$ than the entailment set in $\mathcal{L}'$ of $\mathcal{O}$, where the distance here is measured in terms of set inclusion.

Given an ontology $\mathcal{O}$, it may be the case that the entailment set in a language $\mathcal{L}'$ of $\mathcal{O}$ is infinite. If so, it may happen that the approximation in $\mathcal{L}'$, in accordance with Definition 2, does not exist. For this reason, we follow the approach proposed in [3], where safeness restrictions on the target language are imposed, in order to guarantee the finiteness of the entailment set. For example, in *DL-Lite*$_A$, which is the language we focus on in the following sections, the infiniteness of the entailment set arises from the possibility of inferring infinitely-long existential chains. In this case, the safeness restriction requires to allow, in the TBox, only existential chains of bounded length. Therefore, in what follows we denote versions of *DL-Lite*$_A$ in which such restriction is enforced as *DL-Lite*$_A^{(k)}$, where k is the bounded length of the existential chains.

Another characteristic of *DL-Lite*$_A$, shared with other languages such as $\mathcal{EL}^{++}$ [1], is that syntactic restrictions are imposed on the manner in which assertions can be combined. In this case, there may exist more than one ontology which is an approximation in $\mathcal{L}'$ of $\mathcal{O}$. In what follows we denote with $Apx_{MAX}(\mathcal{O}, \mathcal{L}')$ the set of $\mathcal{L}'$ ontologies which are approximations in $\mathcal{L}'$ of $\mathcal{O}$ in accordance with Definition 2.

Example 1. Consider the OWL 2 ontology $\mathcal{O} = \{A \sqsubseteq \exists R.B, A \sqsubseteq \forall R.B, (\mathsf{funct}\ R)\}$. It is easy to see that, due to the syntactic restriction in *DL-Lite*$_A$, which imposes that a non primitive role cannot be functional, we have that, up to logical equivalence, the set $Apx_{MAX}(\mathcal{O}, \textit{DL-Lite}_A^{(1)}) = \{\{A \sqsubseteq \exists R.B\}, \{A \sqsubseteq \exists R, (\mathsf{funct}\ R)\}\}$. □

Comparison with related work. The problem of approximating ontologies for OBDA applications has recently been faced in [12] and [3].

In [12], the authors define the approximation in $\mathcal{L}'$ of a satisfiable ontology $\mathcal{O}$ as the set of $\mathcal{L}'$ axioms coinciding with the entailment set of $\mathcal{O}$ with respect to $\mathcal{L}'$.

Definition 3. *Let $\mathcal{O}$ be a satisfiable ontology expressed in a language $\mathcal{L}$. The approximation in $\mathcal{L}'$ of $\mathcal{O}$ is $Apx_{ES}(\mathcal{O}, \mathcal{L}') = \mathsf{ES}(\mathcal{O}, \mathcal{L}')$.*

Therefore, in accordance with the above definition, the approximation in $\mathcal{L}'$ of the ontology $\mathcal{O}$ is a set of $\mathcal{L}'$ axioms, but not necessarily a valid ontology expressed in $\mathcal{L}'$.

In [3], the authors provide a more sophisticated notion of approximation, in which it is required that the approximation in $\mathcal{L}'$ of $\mathcal{O}$ be an ontology expressed in $\mathcal{L}'$.

Definition 4. *Let $\mathcal{O}$ be a satisfiable ontology expressed in a language $\mathcal{L}$ over a signature Σ. A satisfiable $\mathcal{L}'$ ontology $\mathcal{O}'$ over Σ is an approximation in $\mathcal{L}'$ of $\mathcal{O}$ if both the following statements hold:*

(i) *$\mathsf{ES}(\mathcal{O}',\mathcal{L}') \subseteq \mathsf{ES}(\mathcal{O},\mathcal{L}')$;*
(ii) *for each $\alpha \in \mathsf{ES}(\mathcal{O},\mathcal{L}')$ such that $\mathcal{O}' \cup \{\alpha\}$ is an $\mathcal{L}'$ ontology, then $\alpha \in \mathsf{ES}(\mathcal{O}',\mathcal{L}')$.*

In other words, an $\mathcal{L}'$ ontology $\mathcal{O}'$ is an approximation in $\mathcal{L}'$ of $\mathcal{O}$ if among all the possible maximal subsets of $\mathsf{ES}(\mathcal{O},\mathcal{L}')$ which are $\mathcal{L}'$ ontologies, there is one which is logically equivalent to $\mathcal{O}'$. It is not unexpected, as with our approach, that there may exist more than one ontology which is an approximation in $\mathcal{L}'$ of $\mathcal{O}$. We denote with $Apx_{SC}(\mathcal{O},\mathcal{L}')$ set of $\mathcal{L}'$ ontologies which are all approximations in $\mathcal{L}'$ of $\mathcal{O}$ in accordance with Definition 4.

Differently from Definition 3, Definition 4 guarantees that an approximation is always an ontology expressed in the target language $\mathcal{L}'$.

Similarly to our notion of approximation, if the entailment set is infinite, it may happen that there is no ontology expressed in the target language which is an approximation. Hence, in order to guarantee the existence of an approximation, one must impose safeness restrictions on the target language in this case as well, in order to guarantee the finiteness of the entailment set.

We now compare our approach to those in [3] and [12] by means of some examples.

Example 2. Consider the following OWL 2 ontology: $\mathcal{O} = \{\exists R^- \sqsubseteq B,\ A \sqsubseteq \exists R.B,\ (\mathsf{funct}\ R)\}$. In accordance with Definitions 3, 4, 2, we have:

$$\begin{aligned}
Apx_{ES}(\mathcal{O},\textit{DL-Lite}_A^{(1)}) = \{&\exists R^- \sqsubseteq B,\ A \sqsubseteq \exists R,\ A \sqsubseteq \exists R.B,\ A \sqsubseteq \exists R.\exists R^-,\\
&\exists R \sqsubseteq \exists R.\exists R^-,\ \exists R^- \sqsubseteq \exists R^-.\exists R,\ (\mathsf{funct}\ R)\ \}
\end{aligned}$$

$$\begin{aligned}
Apx_{SC}(\mathcal{O},\textit{DL-Lite}_A^{(1)}) = \{&\mathcal{O}'_{sc} = \{\exists R^- \sqsubseteq B,\ A \sqsubseteq \exists R,\ A \sqsubseteq \exists R.B, A \sqsubseteq \exists R.\exists R^-,\\
&\qquad\exists R \sqsubseteq \exists R.\exists R^-,\ \exists R^- \sqsubseteq \exists R^-.\exists R\ \},\\
&\mathcal{O}''_{sc} = \{\exists R^- \sqsubseteq B,\ A \sqsubseteq \exists R,\ \ (\mathsf{funct}\ R)\ \}\ \}
\end{aligned}$$

$$Apx_{MAX}(\mathcal{O},\textit{DL-Lite}_A^{(1)}) = \{\mathcal{O}_{max} = \{\exists R^- \sqsubseteq B,\ A \sqsubseteq \exists R,\ \ (\mathsf{funct}\ R)\ \}\ \} \qquad \square$$

Due to the syntactic restrictions enforced in *DL-Lite*$_A^{(1)}$, ontology $\mathcal{O}$ of Example 2 is not a *DL-Lite*$_A^{(1)}$ ontology. However, it is logically equivalent to the *DL-Lite*$_A^{(1)}$ ontology $\{\exists R^- \sqsubseteq B, A \sqsubseteq \exists R, (\mathsf{funct}\ R)\}$.

Regarding $Apx_{ES}(\mathcal{O},\textit{DL-Lite}_A^{(1)})$ we only observe that it is not a valid *DL-Lite*$_A^{(1)}$ ontology. Up to logical equivalence, we can see that the set $Apx_{SC}(\mathcal{O},\textit{DL-Lite}_A^{(1)})$ contains, along with ontology $\mathcal{O}''_{sc}$, which is logically equivalent to $\mathcal{O}$, also the unexpected ontology $\mathcal{O}'_{sc}$, for which we have $\mathsf{ES}(\mathcal{O}'_{sc},\textit{DL-Lite}_A^{(1)}) \subset \mathsf{ES}(\mathcal{O}''_{sc},\textit{DL-Lite}_A^{(1)})$. Finally, according to Definition 2, up to logical equivalence, the only approximation is a *DL-Lite*$_A^{(1)}$ ontology that is logically equivalent to $\mathcal{O}$.

Other significant differences between these three approaches arise when the goal is to compute the approximation of an ontology $\mathcal{O}$ expressed in a language $\mathcal{L}$ in which the UNA is not adopted into a target language $\mathcal{L}'$ in which the UNA is adopted. In the example below, we highlight the different behavior of the three approaches in this circumstance.

Example 3. We recall that in OWL 2 the UNA is not adopted. This means, for instance, that given the following two ontologies $\mathcal{O} = \{A \sqsubseteq \{o_1\}, B \sqsubseteq \{o_2\}\}$ and $\mathcal{O}' = \{A \sqsubseteq$

$\{o_1\}, B \sqsubseteq \{o_2\}, o_1 \neq o_2\}$, we have that $Mod(\mathcal{O}) \neq Mod(\mathcal{O}')$. In fact, while $\mathcal{O}$ does not entail $A \sqsubseteq \neg B$, $\mathcal{O}'$ does. Differently, if we adopt the UNA in OWL 2, then we have that $Mod(\mathcal{O}) = Mod(\mathcal{O}')$, and both entail the assertion $A \sqsubseteq \neg B$.

From the observations above, it follows that $\{A \sqsubseteq \{o_1\}, B \sqsubseteq \{o_2\}\} \subseteq \mathsf{ES}(\mathcal{O}, \text{OWL}_{UNA})$, and that $A \sqsubseteq \neg B \notin \mathsf{ES}(\mathcal{O}, \text{OWL}_{UNA})$. In what follows let us denote with OWL$_{UNA}$ the version of OWL 2 where the UNA is adopted.

In accordance with Definitions 3, 4, and 2, we have that, up to logical equivalence, the approximations in OWL$_{UNA}$ of $\mathcal{O}$ are:

- $Apx_{ES}(\mathcal{O}, \text{OWL}_{UNA}) = \mathsf{ES}(\mathcal{O}, \text{OWL}_{UNA})$. In this case, $Apx_{ES}(\mathcal{O}, \text{OWL}_{UNA})$ is a valid OWL$_{UNA}$ ontology. Let $\mathcal{O}_{ES}$ be such ontology. As shown above, since in OWL$_{UNA}$ the UNA is adopted, $\mathcal{O}_{ES} \models A \sqsubseteq \neg B$. This means that $\mathsf{ES}(\mathcal{O}_{ES}, \text{OWL}_{UNA}) \not\subseteq \mathsf{ES}(\mathcal{O}, \text{OWL}_{UNA})$.
- $Apx_{SC}(\mathcal{O}, \text{OWL}_{UNA}) = \emptyset$. Indeed, since $\mathsf{ES}(\mathcal{O}, \text{OWL}_{UNA})$ is a valid OWL$_{UNA}$ ontology, any OWL$_{UNA}$ ontology $\mathcal{O}_1$ such that $\mathsf{ES}(\mathcal{O}_1, \text{OWL}_{UNA}) \subset \mathsf{ES}(\mathcal{O}, \text{OWL}_{UNA})$, does not satisfy condition (ii) of Definition 4. Moreover, since every OWL$_{UNA}$ ontology $\mathcal{O}_2$ that satisfies condition (ii) entails both $\{A \sqsubseteq \{o_1\}, B \sqsubseteq \{o_2\}\}$, then it also entails $A \sqsubseteq \neg B$. Hence $\mathsf{ES}(\mathcal{O}_2, \text{OWL}_{UNA}) \not\subseteq \mathsf{ES}(\mathcal{O}, \text{OWL}_{UNA})$, and so condition (i) of Definition 4 is not satisfied.
- $Apx_{MAX}(\mathcal{O}, \text{OWL}_{UNA}) = \{\mathcal{O}'_{max} = \{A \sqsubseteq \{o_1\}\},\ \mathcal{O}''_{max} = \{B \sqsubseteq \{o_2\}\}\ \}$. Indeed, it is easy to verify that both $\mathcal{O}'_{max}$ and $\mathcal{O}''_{max}$ satisfy both conditions (i) and (ii) of Definition 2, and that there is no other ontology $\mathcal{O}'''$ in $Apx_{MAX}(\mathcal{O}, \text{OWL}_{UNA})$ logically equivalent to neither $\mathcal{O}'_{max}$ nor $\mathcal{O}''_{max}$. □

As shown in the previous example, given an $\mathcal{L}$ ontology $\mathcal{O}$ and a target language $\mathcal{L}'$, our approach, as the one in [3], but differently from the one given in [12], guarantees that an approximation in $\mathcal{L}'$ of $\mathcal{O}$ is an $\mathcal{L}'$ ontology, and that for each $\mathcal{L}'$ ontology $\mathcal{O}'$ that is an approximation in $\mathcal{L}'$ of $\mathcal{O}$, we have that $\mathsf{ES}(\mathcal{O}', \mathcal{L}') \subseteq \mathsf{ES}(\mathcal{O}, \mathcal{L}')$. Finally, it can be shown that our approach always preserves the same or more inferences than those obtained by adopting the approach given in [3].

Theorem 1. *For each $\mathcal{O}_{sc} \in Apx_{SC}(\mathcal{O}, \mathcal{L}')$, there is an $\mathcal{O}_{max} \in Apx_{MAX}(\mathcal{O}, \mathcal{L}')$ such that $\mathsf{ES}(\mathcal{O}_{sc}, \mathcal{L}') \subseteq \mathsf{ES}(\mathcal{O}_{max}, \mathcal{L}')$. Furthermore, for each $\mathcal{O}_{max} \in Apx_{MAX}(\mathcal{O}, \mathcal{L}')$, there is no $\mathcal{O}_{sc} \in Apx_{SC}(\mathcal{O}, \mathcal{L}')$ such that $\mathsf{ES}(\mathcal{O}_{max}, \mathcal{L}') \subset \mathsf{ES}(\mathcal{O}_{sc}, \mathcal{L}')$.*

4 Approximation in *DL-Lite$_A$* of OWL 2 ontologies

In this section, we study the problem of computing the approximation in *DL-Lite$_A$* of a satisfiable OWL 2 ontology $\mathcal{O}$. More specifically, we aim to approximate $\mathcal{O}$ with *DL-Lite$_A$* TBox assertions. Therefore, in what follows we assume that $Apx_{MAX}(\mathcal{O}, \textit{DL-Lite}_A)$ is a set of *DL-Lite$_A$* TBoxes and that $\mathsf{ES}(\mathcal{O}, \textit{DL-Lite}_A)$ is a set of *DL-Lite$_A$* TBox assertions. To guarantee the finiteness of the entailment set, we refer to versions of *DL-Lite$_A$* allowing only for TBox assertions with existential chains of bounded length k.

Given a set of *DL-Lite*$_A^{(k)}$ assertions $\mathcal{S}$, and a functionality assertion φ over a role R (resp. attribute U), we denote with $clashes(\varphi, \mathcal{S})$ the set of all assertions involving R (resp. U) that, together with φ, violate the syntactic restriction imposed on *DL-Lite*$_A^{(k)}$ TBoxes. Hence, $clashes(\varphi, \mathcal{S})$ is a set of role (resp. attribute) inclusion assertions and assertions with a qualified existential role (resp. attribute) on the right hand side.

Let $\mathcal{O}$ be an OWL 2 ontology, and let $\mathcal{F}$ be the set containing all the functionality assertions in $\mathsf{ES}(\mathcal{O}, \textit{DL-Lite}_A^{(k)})$ for which $clashes(\varphi, \mathsf{ES}(\mathcal{O}, \textit{DL-Lite}_A^{(k)})) \neq \emptyset$. If $\mathcal{F} \neq \emptyset$, then $\mathsf{ES}(\mathcal{O}, \textit{DL-Lite}_A^{(k)})$ is not a valid *DL-Lite*$_A^{(k)}$ TBox, and there are at most $2^{|\mathcal{F}|}$ TBoxes $\mathcal{T}_i$ which are valid *DL-Lite*$_A^{(k)}$ TBoxes and minimally differ from $\mathsf{ES}(\mathcal{O}, \textit{DL-Lite}_A^{(k)})$. Let $MaxSub_{ES}(\mathsf{ES}(\mathcal{O}, \textit{DL-Lite}_A^{(k)}))$ be the set of such *DL-Lite*$_A^{(k)}$ TBoxes. One can compute these TBoxes in $MaxSub_{ES}(\mathsf{ES}(\mathcal{O}, \textit{DL-Lite}_A^{(k)}))$ by retracting, from $\mathsf{ES}(\mathcal{O}, \textit{DL-Lite}_A^{(k)})$, either $\varphi \in \mathcal{F}$ or the assertions in $clashes(\varphi, \mathcal{S})$, in order to resolve the violations of the syntactic restriction.

The lemma below guarantees that a TBox in $MaxSub_{ES}(\mathsf{ES}(\mathcal{O}, \textit{DL-Lite}_A^{(k)}))$ is a candidate for being one of the TBoxes in $Apx_{MAX}(\mathcal{O}, \textit{DL-Lite}_A^{(k)})$.

Lemma 1. *Let $\mathcal{O}$ be a satisfiable OWL 2 ontology and let $\mathcal{T}$ be a DL-Lite$_A^{(k)}$ TBox. $\mathsf{ES}(\mathcal{T}, \textit{DL-Lite}_A^{(k)}) \subseteq \mathsf{ES}(\mathcal{O}, \textit{DL-Lite}_A^{(k)})$ if and only if $\mathcal{T} \subseteq \mathsf{ES}(\mathcal{O}, \textit{DL-Lite}_A^{(k)})$.*

In other words, the above lemma guarantees that the first condition in Definition 2 is satisfied by every *DL-Lite*$_A^{(k)}$ TBox in $MaxSub_{ES}(\mathsf{ES}(\mathcal{O}, \textit{DL-Lite}_A^{(k)}))$, and that in computing all the TBoxes in $Apx_{MAX}(\mathcal{O}, \textit{DL-Lite}_A^{(k)})$, one can consider only the assertions in $\mathsf{ES}(\mathcal{O}, \textit{DL-Lite}_A^{(k)})$. Moreover, from the monotonicity of the DLs, it directly follows that for every TBox $\mathcal{T}$ in $Apx_{MAX}(\mathcal{O}, \textit{DL-Lite}_A^{(k)})$ there is a TBox in $MaxSub_{ES}(\mathsf{ES}(\mathcal{O}, \textit{DL-Lite}_A^{(k)}))$ which is logically equivalent to $\mathcal{T}$.

However, in order for a TBox $\mathcal{T}_i$ in $MaxSub_{ES}(\mathsf{ES}(\mathcal{O}, \textit{DL-Lite}_A^{(k)}))$ to belong to $Apx_{MAX}(\mathcal{O}, \textit{DL-Lite}_A^{(k)})$, it must also satisfy the second condition of Definition 2, and thus that there is no other *DL-Lite*$_A^{(k)}$ TBox $\mathcal{T}' \subseteq \mathsf{ES}(\mathcal{T}, \textit{DL-Lite}_A^{(k)})$ such that $\mathsf{ES}(\mathcal{T}_i, \textit{DL-Lite}_A^{(k)}) \subset \mathsf{ES}(\mathcal{T}', \textit{DL-Lite}_A^{(k)}) \subseteq \mathsf{ES}(\mathcal{O}, \textit{DL-Lite}_A^{(k)})$.

To explain why a TBox in $MaxSub_{ES}(\mathsf{ES}(\mathcal{O}, \textit{DL-Lite}_A^{(k)}))$ does not necessarily also satisfy the second condition in Definition 2, we refer to the ontology $\mathcal{O} = \{\exists R^- \sqsubseteq B, A \sqsubseteq \exists R.B, (\mathsf{funct}\ R)\}$ of Example 2. It is easy to see that $\mathsf{ES}(\mathcal{O}, \textit{DL-Lite}_A^{(1)}) = \{\exists R^- \sqsubseteq B,\ A \sqsubseteq \exists R.B,\ (\mathsf{funct}\ R),\ \exists R \sqsubseteq \exists R.\exists R^-,\ \exists R \sqsubseteq \exists R.B,\ A \sqsubseteq \exists R.\exists R^-,\ \exists R^- \sqsubseteq \exists R^-.\exists R,\ A \sqsubseteq \exists R\}$, and that $clashes((\mathsf{funct}\ R), \mathsf{ES}(\mathcal{O}, \textit{DL-Lite}_A^{(1)})) = \{A \sqsubseteq \exists R.B,\ \exists R \sqsubseteq \exists R.\exists R^-,\ \exists R \sqsubseteq \exists R.B,\ A \sqsubseteq \exists R.\exists R^-,\ \exists R^- \sqsubseteq \exists R^-.\exists R\}$. Hence, by following the procedure described above, we have the following two TBoxes in $MaxSub_{ES}(\mathsf{ES}(\mathcal{O}, \textit{DL-Lite}_A^{(1)}))$: $\mathcal{T}_1 = \{\exists R^- \sqsubseteq B,\ A \sqsubseteq \exists R.B,\ \exists R \sqsubseteq \exists R.\exists R^-,\ \exists R \sqsubseteq \exists R.B,\ A \sqsubseteq \exists R.\exists R^-,\ \exists R^- \sqsubseteq \exists R^-.\exists R,\ A \sqsubseteq \exists R\}$, and $\mathcal{T}_2 = \{\exists R^- \sqsubseteq B,\ (\mathsf{funct}\ R),\ A \sqsubseteq \exists R\}$. However, it is clear that only $\mathcal{T}_2 \in Apx_{MAX}(\mathcal{O}, \textit{DL-Lite}_A^{(1)})$. In fact we have that $\mathsf{ES}(\mathcal{T}_1, \textit{DL-Lite}_A^{(1)}) \subset \mathsf{ES}(\mathcal{T}_2, \textit{DL-Lite}_A^{(1)})$.

We provide the algorithm $isApx$ which, given a TBox $\mathcal{T}$ and an ontology $\mathcal{O}$, returns *true* if $\mathcal{T} \in Apx_{MAX}(\mathcal{O}, \textit{DL-Lite}_A^{(k)})$, *false* otherwise. The algorithm proceeds as fol-

Algorithm 1: $isApx(\mathcal{T}, \mathcal{O})$

```
Input: a DL-Lite$_A^{(k)}$ TBox $\mathcal{T}$, a satisfiable OWL 2 ontology $\mathcal{O}$
Output: true or false
begin
    $\mathcal{E} \leftarrow \mathsf{ES}(\mathcal{T}, \textit{DL-Lite}_A^{(k)})$;
    $\mathcal{S} \leftarrow \mathsf{ES}(\mathcal{O}, \textit{DL-Lite}_A^{(k)}) \setminus \mathcal{E}$;
    foreach $\alpha \in \mathcal{S}$
        if $\mathcal{T} \cup \{\alpha\}$ is a DL-Lite$_A^{(k)}$ TBox then return false;
    foreach functionality assertion $\phi \in \mathcal{E}$
        $\mathcal{E} \leftarrow \mathcal{E} \setminus clashes(\phi, \mathcal{E})$;
    foreach functionality assertion $\varphi \in \mathcal{S}$
        if $\mathsf{ES}(\mathcal{E} \setminus clashes(\varphi, \mathcal{E}), \textit{DL-Lite}_A^{(k)}) = \mathsf{ES}(\mathcal{T}, \textit{DL-Lite}_A^{(k)})$ then return false;
return true;
end
```

lows. Given a satisfiable OWL 2 ontology $\mathcal{O}$ and a *DL-Lite*$_A^{(k)}$ TBox $\mathcal{T}$, it first computes their entailment sets in *DL-Lite*$_A^{(k)}$ and then computes the set $\mathcal{S}$ containing all the assertions in the entailment set of $\mathcal{O}$ in *DL-Lite*$_A^{(k)}$ which are not in the entailment set of $\mathcal{T}$ in *DL-Lite*$_A^{(k)}$. Then, for every assertion α in $\mathcal{S}$, it attempts to add α to $\mathcal{T}$ without violating any syntactic restriction. If such assertion exists, then $\mathcal{T}$ is not an approximation in *DL-Lite*$_A^{(k)}$ of $\mathcal{O}$. If not, the algorithm continues, fixing $\mathcal{E}$ as the entailment set of $\mathcal{T}$ in *DL-Lite*$_A^{(k)}$, and resolving every violation of the syntactic restriction in $\mathcal{E}$ by removing $clashes(\phi, \mathcal{E})$ from $\mathcal{E}$, for every functionality assertion ϕ in $\mathcal{E}$. This operation guarantees that $\mathcal{E}$ is a *DL-Lite*$_A^{(k)}$ TBox. Then the algorithm checks, for every functionality assertion φ in $\mathcal{S}$, if the set obtained from $\mathcal{E}$ by removing $clashes(\varphi, \mathcal{E})$ from $\mathcal{E}$ is logically equivalent to $\mathcal{E}$. If this is the case, then it is possible to build a *DL-Lite*$_A^{(k)}$ TBox $\mathcal{T}''$ by adding φ to the set of assertions obtained in this fashion. $\mathcal{T}''$ is a *DL-Lite*$_A^{(k)}$ TBox that proves that $\mathcal{T}$ does not satisfy the second condition of Definition 2. Otherwise, the algorithm terminates by returning *true*.

The theorem below establishes termination and correctness of Algorithm 1.

Theorem 2. *Let $\mathcal{O}$ be a satisfiable OWL 2 ontology, and let $\mathcal{T}$ be a DL-Lite$_A^{(k)}$ TBox. $isApx(\mathcal{T}, \mathcal{O})$ terminates, and returns true if and only if $\mathcal{T} \in Apx_{MAX}(\mathcal{O}, \textit{DL-Lite}_A^{(k)})$.*

Given a satisfiable OWL 2 ontology $\mathcal{O}$, Lemma 1 and Theorem 2 suggest Algorithm 2 for computing, up to logical equivalence, the set $Apx_{MAX}(\mathcal{O}, \textit{DL-Lite}_A^{(k)})$.

The following theorem establishes termination and correctness of Algorithm 2.

Theorem 3. *Let $\mathcal{O}$ be a satisfiable OWL 2 ontology. Then $computeApx(\mathcal{O})$ terminates and computes, up to logical equivalence, $Apx_{MAX}(\mathcal{O}, \textit{DL-Lite}_A^{(k)})$.*

As expected, Algorithm 2 does not return a single TBox, but instead a set of TBoxes. For application purposes, the approximation that shall be used must be chosen from this set. A pragmatic approach could be to choose one non-deterministically. Instead, one could think to leave this choice to the end user, according to the use he intends to make

Algorithm 2: $computeApx(\mathcal{O})$

```
Input: a satisfiable OWL 2 ontology $\mathcal{O}$
Output: a set of $DL\text{-}Lite_A^{(k)}$ TBoxes
begin
    $\mathcal{S} \leftarrow MaxSub_{ES}(\mathsf{ES}(\mathcal{O}, DL\text{-}Lite_A^{(k)}))$;
    foreach $\mathcal{T}_i \in \mathcal{S}$
        if $isApx(\mathcal{T}_i, \mathcal{O}) = false$ then $\mathcal{S} \leftarrow \mathcal{S} \setminus \{\mathcal{T}_i\}$;
return $\mathcal{S}$;
end
```

of it. A more interesting direction could be to achieve the identification of a unique TBox by applying some preference criteria to the set returned by Algorithm 2.

5 Efficient entailment set computation in *DL-Lite*

In the previous sections, we have provided an algorithm for computing the set $Apx_{MAX}(\mathcal{O}, DL\text{-}Lite_A^{(k)})$ of TBoxes for an OWL 2 ontology $\mathcal{O}$. This computation is clearly intractable. Indeed, it requires to compute the set $\mathsf{ES}(\mathcal{O}, DL\text{-}Lite_A^{(k)})$, and moreover, due to the syntactic restriction enforced in $DL\text{-}Lite_A^{(k)}$, in the worst case, the cardinality of the set $Apx_{MAX}(\mathcal{O}, DL\text{-}Lite_A^{(k)})$ is exponential in the number of functionality assertions in $\mathsf{ES}(\mathcal{O}, DL\text{-}Lite_A^{(k)})$. In particular, the computation of $\mathsf{ES}(\mathcal{O}, DL\text{-}Lite_A^{(k)})$ is in general very costly, as highlighted also in [3] and [12], since it requires the invocation of reasoning services over an OWL 2 ontology $\mathcal{O}$. This task is performed by invoking an OWL 2 oracle which can be implemented by an OWL 2 reasoner.

A naive algorithm for computing $\mathsf{ES}(\mathcal{O}, DL\text{-}Lite_A^{(k)})$ is the one described in [12], in which firstly one computes the set Γ of $DL\text{-}Lite_A^{(k)}$ TBox assertions which can be built over the signature Σ, and then, for each assertion $\alpha \in \Gamma$ such that $\mathcal{O} \models \alpha$, one adds α to $\mathsf{ES}(\mathcal{O}, DL\text{-}Lite_A^{(k)})$.

We now show how to optimize the computation of $\mathsf{ES}(\mathcal{O}, DL\text{-}Lite_A^{(k)})$ through a technique which drastically reduces in practice the calls to the OWL 2 oracle.

In the computation of $\mathsf{ES}(\mathcal{O}, DL\text{-}Lite_A^{(k)})$, a large portion of the invocations of the OWL 2 oracle involve assertions in which a general concept $C_{\exists R_1 \ldots \exists R_n}$ involving a non trivial existential role chain occurs. Empirical observation has brought to light the fact that this kind of general concept very often does not subsume any concept in $\mathcal{O}$. Hence, all the invocations of the OWL 2 oracle involving these childless general concepts are useless. Therefore, at the base of our strategy is the identification of all these childless general concepts $C_{\exists R_1 \ldots \exists R_n}$, without invoking the OWL 2 oracle.

We use the function $subsumed(S_1, O)$, where S_1 is a general concept (resp. general role, general attribute) which returns the set of concepts (resp. roles, attributes) S_2 such that $\mathcal{O} \models \mathcal{S}_2 \sqsubseteq S_1$. This function is efficiently performed by the most commonly-used OWL 2 reasoners, such as Pellet [15], Racer [8], FACT++ [16], and HermiT [7].

Our technique calls, as the first step, for the computation of the classification of basic concepts, roles, and attributes, which is encoded into a directed graph, in which the nodes represent the predicates of the ontology, and the edges the inclusion assertions.

Ontology	DL Fragment	#O.A.			#N.A.			Total computation time (ms)		
		k = 1	k = 2	k = 3	k = 1	k = 2	k = 3	k = 1	k = 2	k = 3
Mouse	$\mathcal{ALCI}$	6.059	12.611	19.163	11.018	40.362	157.738	8.426	9.955	12.173
Pathway	$\mathcal{EL}$	10.191	11.999	11.999	14.294	52.374	204.694	11.975	16.498	17.553
Cognitive	$\mathcal{SHIF}(D)$	56.883	178.381	474.145	48.006	541.350	6.461.478	348.892	1.812.511	6.832.865
Mammalian	$\mathcal{AL}+$	7.551	7.551	7.551	112.898	413.922	1.618.018	322.527	350.853	350.853
Spatial	$\mathcal{ALEHI}$	51.065	82.735	150.195	47.143	4.541.815	445.019.671	27.827	52.742	132.807

Table 1: Evaluation of the optimized algorithm for computing $\mathsf{ES}(\mathcal{O}, \textit{DL-Lite}_A^{(k)})$.

After this initial step, the remaining invocations, which we work to minimize, are those needed for computing the entailed inclusion assertions involving general concepts $C_{\exists R_1 \ldots \exists R_n}$, and the entailed disjointness. Regarding the former, we exploit the graph encoding of concept, role, and attribute classification to invoke these subsumption checks in a manner which follows the hierarchical order of these general concepts $C_{\exists R_1 \ldots \exists R_n}$, in order to avoid those checks which can be skipped. Consider, for example, an ontology $\mathcal{O}$ that entails the inclusions $A_1 \sqsubseteq A_2$ and $P_1 \sqsubseteq P_2$, where A_1 and A_2 are concepts and P_1 and P_2 are roles. Exploiting these inclusions we can deduce the hierarchical structure involving general concepts that can be built on these four predicates. For instance, we know that $\exists P_2.A_2 \sqsubseteq \exists P_2$, that $\exists P_2.A_1 \sqsubseteq \exists P_2.A_2$, that $\exists P_1.A_1 \sqsubseteq \exists P_2.A_1$, and so on. We begin by invoking the OWL 2 oracle, asking for the children of the general concepts which are in the highest position in the hierarchy. So, first we call $subsumed(\exists P_2, \mathcal{O})$. If $subsumed(\exists P_2, \mathcal{O}) = \emptyset$, we avoid invoking the oracle asking for $subsumed(\exists P_2.A_2, \mathcal{O})$, and so on. Regarding the latter we follow the same procedure, but beginning from the lowest positions in the hierarchy. The algorithm concludes by asking the OWL 2 oracle for all functionalities entailed by $\mathcal{O}$.

In Table 1 we present a sample of the evaluation tests for this strategy, performed through a Java-based implementation of this technique. The table reports the number of invocations to the OWL 2 oracle performed with our optimization (#O.A.), and without (#N.A.), for computing the entailment set of the ontologies in *DL-Lite*$_A^{(k)}$, with $1 \leq k \leq 3$. It also reports, for each ontology, the total computation time of $\mathsf{ES}(\mathcal{O}, \textit{DL-Lite}_A^{(k)})$.

6 Conclusion

In this paper we address the problem of ontology approximation. We illustrate our approach to this problem, and provide a comparison with other approaches provided in literature. We deeply investigate the approximation of OWL 2 ontologies with *DL-Lite* TBoxes for OBDA purposes, and provide an algorithm for its computation. Finally, we present a technique for the optimization of the core procedure of this computation, whose success we have shown with empirical evaluation. As future work, we plan to improve the performances in computing the approximation in *DL-Lite* of OWL 2 ontologies by adopting more sophisticated techniques. Moreover, we aim to study reasonable solutions for addressing the problem of multiple approximations of an ontology. In particular, for those settings in which the approximation is used in OBDA.

Acknowledgments. This research has been partially supported by the EU under FP7 project Optique (grant n. FP7-318338), and by the EU under FP7-ICT project ACSI (grant no. 257593).

References

1. Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the $\mathcal{EL}$ envelope further. In *Proc. of OWLED 2008 DC*, 2008.
2. Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2003.
3. Elena Botoeva, Diego Calvanese, and Mariano Rodriguez-Muro. Expressive approximations in DL-Lite ontologies. *Proc. of AIMSA 2010*, pages 21–31, 2010.
4. Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Antonella Poggi, Mariano Rodriguez-Muro, Riccardo Rosati, Marco Ruzzi, and Domenico Fabio Savo. The Mastro system for ontology-based data access. *Semantic Web J.*, 2(1):43–53, 2011.
5. Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *J. of Automated Reasoning*, 39(3):385–429, 2007.
6. Balakrishnan Chandrasekaran, John R Josephson, and V Richard Benjamins. What are ontologies, and why do we need them? *Intelligent Systems and Their Applications, IEEE*, 14(1):20–26, 1999.
7. Birte Glimm, Ian Horrocks, Boris Motik, Rob Shearer, and Giorgos Stoilos. A novel approach to ontology classification. *J. of Web Semantics*, 10(1), 2011.
8. Volker Haarslev and Ralf Möller. RACER system description. In *Proc. of IJCAR 2001*, volume 2083 of *LNAI*, pages 701–705. Springer, 2001.
9. Pascal Hitzler, Markus Krötzsch, Bijan Parsia, Peter F. Patel-Schneider, and Sebastian Rudolph, editors. *OWL 2 Web Ontology Language: Primer*. W3C Recommendation, 11 december 2012. Available at `http://www.w3.org/TR/2012/REC-owl2-primer-20121211/`.
10. Maurizio Lenzerini. Ontoloogy-based data management. In *Proc. of CIKM 2011*, pages 5–6, 2011.
11. Boris Motik, Bijan Parsia, and Peter F. Patel-Schneider, editors. *OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax*. W3C Recommendation, 11 december 2012. Available at `http://www.w3.org/TR/2012/REC-owl2-syntax-20121211/`.
12. Jeff Z Pan and Edward Thomas. Approximating OWL-DL ontologies. In *Proc. of AAAI 2007*, page 1434, 2007.
13. Antonella Poggi, Domenico Lembo, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. Linking data to ontologies. *J. on Data Semantics*, X:133–173, 2008.
14. Bart Selman and Henry Kautz. Knowledge compilation and theory approximation. *J. of the ACM*, 43(2):193–224, 1996.
15. Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. Pellet: a practical OWL-DL reasoner. *Web Semantics: science, services and agents on the World Wide Web*, 5(2):51–53, 2007.
16. Dmitry Tsarkov and Ian Horrocks. FaCT++ description logic reasoner: System description. In *Proc. of IJCAR 2006*, pages 292–297, 2006.
17. Tuvshintur Tserendorj, Sebastian Rudolph, Markus Krötzsch, and Pascal Hitzler. Approximate OWL-reasoning with Screech. In *Proc. of RR 2008*, pages 165–180. Springer, 2008.
18. Holger Wache, Perry Groot, and Heiner Stuckenschmidt. Scalable instance retrieval for the semantic web by approximation. In *Proc. of WISE 2005*, pages 245–254. Springer, 2005.

Appendix J

Empirical Evaluation of the Ontology Approximation Module

In what follows, we present the experimental tests that have been performed to evaluate the efficiency of the optimization technique presented in the paper in Appendix I for the computation of the entailment set of an OWL 2 ontology in OWL 2 QL. The purpose of these tests is to compare the number of invocations to the OWL 2 reasoner that are executed when adopting the optimized approach presented in the paper, and when adopting the naive approach [56].

The suite of ontologies used during testing contains more than twenty ontologies and was assembled from the Bioportal ontology repository[1]. The ontologies that compose this suite are selected to test the scalability of this approach both to larger ontologies and to ontologies formulated in more expressive languages. In Table J.1 we present the most relevant metrics of these ontologies.

All tests were performed on a DELL Latitude E6320 notebook with Intel Core i7-2640M 2.8Ghz CPU and 4GB of RAM, running Microsoft Windows 7 Premium operating system, and Java 1.6 with 2GB of heap space. Timeout was set at two hours, and execution was aborted if maximum available memory was exhausted.

In Table J.2 we present the results of the evaluation conducted using the Pellet reasoner. Total computation times are in milliseconds. The results of the experimentation show that this technique is successful in reducing the number of invocations of the OWL 2 oracle for all tested ontologies.

The paper in Appendix H presents an empirical evaluation of the approximation approach presented in Section 5.5.

As one can notice, by choosing a value for k different from $|\mathcal{O}_S|$, the computation of the entailment set becomes easier. However, observing Algorithm 2, the number of times that this step must be repeated can grow very quickly. In fact, the number of sets of axioms in $subset_k(\mathcal{O}_S)$ is equal to the binomial coefficient of $|\mathcal{O}_S|$ over k, and therefore for large ontologies this number can easily become enormous, and this can be in practice a critical obstacle in the computation of the k-approximation.

For this reason, the experiments presented in the paper focus on comparing the GSA (k-approximation with $k = |\mathcal{O}_S|$) to the LSA (k-approximation with $k = 1$). Furthermore, a syntactic sound approximation approach is included in the evaluation, in order to provide a standard baseline against which to compare the results of the GSA and LSA. This syntactic approximation consists in first normalizing the axioms in the ontology and then eliminating the ones that are not syntactically compliant with OWL 2 QL.

The suite of ontologies used during testing was assembled from the Bioportal ontology repository. The ontologies that compose this suite were selected to test the scalability of our approaches both to larger ontologies and to ontologies formulated in more expressive languages. The implementation of the GSA and LSA used in these experiments adopts the optimization for the computation of the entailment set presented in the previous section.

For the complete results of the experiments we refer the reader to Table 2 of the paper in Appendix H. As one can see, these results An analysis of these results leads to the following observations.

First of all, one can observe that it was possible to compute the GSA only for 26 out of the 41 tested ontologies. The LSA was instead always computed, was quicker than the GSA approach for all but one of the tested ontologies, and was overall very fast.

Secondly, it is interesting to observe the comparison between the quality of the approximation that one can obtain through the LSA with respect to that obtained through the GSA. This relationship answers the question of whether the ontology obtained through the LSA (the “LSA ontology") is able to capture a significant portion of the one obtained

[1]`http://bioportal.bioontology.org/`

Ontology	DL fragment	Concepts	Roles	Attr.	Positive inclusions	Negative inclusions	Max depth	Max siblings
Vertebrate Anat.	*SHIF*	43	14	0	58	46	9	9
Protein	*ALCF(D)*	45	50	133	455	69	6	8
Amino Acid	*ALCF(D)*	46	5	1	261	199	3	9
Biopax	*ALCF(D)*	69	55	41	322	13	5	15
Patient Record	*ALCHI*	89	37	1	132	38	12	7
Cog Analysis	*SHIF(D)*	92	37	9	174	2	11	27
Diagnostic	*ALCOF(D)*	96	4	5	121	121	6	9
Pizza	*SHOIN*	100	8	0	291	398	6	23
Time Event	*ALCOIQ(D)*	104	28	7	191	14	10	35
Spatial	*ALEHI*	136	49	0	230	0	2	32
Translational Medicine	*ALCIF(D)*	225	18	6	259	23	12	14
Anatomical Ent.	*EL*	250	13	0	368	0	11	13
Physical Exer.	*ALCHIF(D)*	634	18	9	670	5	9	103
Pediatric	*AL*	886	0	3	893	0	9	50
Mouse Brain	*ALCI*	913	2	0	915	2525	10	28
Pathway	*EL*	1186	2	0	1469	0	9	27
Cognitive Atlas	*ALC*	1701	6	0	4010	0	7	41
Mosquito Anat.	*EL*	1864	5	0	2732	0	13	103
Mouse Anatomy	*EL+*	3020	2	0	3827	0	17	128
Fly Tax.	*AL*	6599	0	0	6587	0	35	207
Worm Anat.	*EL*	7201	12	0	12342	0	11	1025
Mammalian Phen.	*AL+*	9403	2	0	11095	0	15	40
Galen	*ALEHIF*	23141	950	0	36964	0	25	1492
Gene	*SH*	39160	12	0	73937	3	18	726
FMA 3.1	*ALCOIN(D)*	83284	122	63	86941	0	21	219

Table J.1: Metrics of the ontologies selected for evaluation of the optimization technique for the computation of an entailment set. The Max depth and Max siblings fields indicate respectively the largest depth of a concept hierarchy in the ontology, and the maximum number of children of any one concept.

through the GSA (the “GSA ontology"). Our tests in fact confirm that this is the case: out of the 26 ontologies for which we were able to compute the GSA, in only four cases the LSA ontology entails less than 60 percent of the axioms of the GSA ontology, while in twenty cases it entails more than 90 percent of them. The average percentage of axioms in the original ontologies entailed by the GSA ontologies is roughly 80 percent, and of the axioms of the GSA ontologies entailed by the LSA ontologies is roughly 87 percent.

Furthermore, the LSA provides a good approximation even for those ontologies for which the GSA is not computable. In fact, Table 3 of the paper in Appendix H shows the percentage of axioms of the original ontology that are entailed by the LSA ontology. Out of the twelve ontologies for which we were able to obtain this value (the remaining three ontologies caused an “out of memory" error), only in three cases it was less than 60 percent, while in four cases it was higher than 80 percent. These results are particularly interesting with respect to those ontologies for which the GSA approach is not feasible due to their complexity, as is the case for example for the Dolce ontology, for Galen-A, and for the Ontology of physics for biology. Indeed, even though these ontologies are expressed in highly expressive DL languages, the structure of the axioms that compose them is such that reasoning on each of them in isolation does not lead to much worse approximation results than reasoning on the ontology as a whole: for the nine smallest ontologies in Table 3 of the paper in Appendix H, for which the GSA fails not because of the size of the ontology, the average percentage is 68.6.

Finally, both the GSA and LSA compare favorably against the syntactic sound approximation approach. In fact, the average percentage of axioms in the LSA and GSA ontologies that are entailed by the ontologies obtained through the SYNT approach are respectively roughly 90 percent and 72 percent. While the latter result is to be expected, the former is quite significant, even more so when one considers that the LSA is very fast. Indeed, a “gain" of 10 percent of axiom entailments by the LSA with respect to the SYNT in the case of large ontologies such as Galen and Snomed translates to tens of thousands of preserved axioms in very little computation time.

In conclusion, the results gathered from these tests corroborate the usefulness of both the global semantic approximation and the local semantic approximation approaches. The former provides a maximal sound approximation in the target language of the original approach, and is in practice computable in a reasonable amount of time for the majority of the tested ontologies. The latter instead represents a less optimal but still very effective solution for those ontologies for which the GSA approach goes beyond the capabilities of the currently-available ontology reasoners.

Ontology	# O.A.I.	# N.A.I.	Total time (ms)
Vertebrate Anat.	1531	4358	1570
Protein	34043	57461	19262
Amino Acid	730	1406	2126
Biopax	37069	49788	13187
Patient Record	17029	25190	5838
Cog Analysis	10945	26886	5566
Diagnostic	1802	2223	1778
Pizza	2309	4232	9199
Time Event	8766	19589	11020
Spatial	51065	77143	27827
Translational Medicine	10407	20436	15795
Anatomical Ent.	11946	15547	5275
Physical Exer.	24645	51757	14922
Pediatric	2495	14293	2999
Mouse Brain	6059	11018	8426
Pathway	10191	14294	11975
Cognitive Atlas	56883	88006	348892
Mosquito Anat.	63712	95011	4066457
Mouse Anatomy	39614	46302	2989335
Fly Taxonomy	18233	26396	582876
Worm Anat.	508805	875784	4565350
Mammalian Phen.	75551	112898	322527
Galen	——	95262614	timeout
Gene	——	2037652	timeout
FMA 3.1	——	41127815	timeout

Table J.2: Evaluation of the optimization algorithm for the computation of $\mathsf{ES}(\mathcal{O}, OWL\ 2\ QL)$ with the Pellet OWL 2 reasoner. # O.A.I. = number of Pellet reasoner invocations by optimized algorithm, # N.A.I. = number of Pellet reasoner invocations by non-optimized algorithm. Total time of $\mathsf{ES}(\mathcal{O}, OWL\ 2\ QL)$ computation expressed in milliseconds.

Appendix K

Optique demo: ISWC 2013 paper

This appendix reports the paper:

- E. Kharlamov, M. Giese, E. Jiménez-Ruiz, M. G. Skjæveland, A. Soylu, D. Zheleznyakov, T. Bagosi, M. Console, P. Haase, I. Horrocks, S. Marciuska, C. Pinkel, M. Rodriguez-Muro, M. Ruzzi, V. Santarelli, D. F. Savo, K. Sengupta, M. Schmidt, E. Thorstensen, J. Trame, and A. Waaler. Optique 1.0: Semantic Access to Big Data: The Case of Norwegian Petroleum Directorate's FactPages. ISWC Demos 2013.

Optique 1.0: Semantic Access to Big Data*
The Case of Norwegian Petroleum Directorate's FactPages

E. Kharlamov[1,**], M. Giese[2], E. Jiménez-Ruiz[1], M. G. Skjæveland[2], A. Soylu[2], D. Zheleznyakov[1], T. Bagosi[3], M. Console[5], P. Haase[4], I. Horrocks[1], S. Marciuska[3], C. Pinkel[4], M. Rodriguez-Muro[3], M. Ruzzi[5], V. Santarelli[5], D. F. Savo[5], K. Sengupta[4], M. Schmidt[4], E. Thorstensen[2], J. Trame[4], and A. Waaler[2]

[1] University of Oxford, UK; [2] University of Oslo, Norway;
[3] Free University of Bozen-Bolzano, Italy; [4] fluid Operations AG, Germany;
[5] Sapienza Università di Roma, Italy

Abstract. The Optique project aims at developing an end-to-end system for semantic data access to Big Data in industries such as Statoil ASA and Siemens AG. In our demonstration we present the first version of the Optique system customised for the Norwegian Petroleum Directorate's FactPages, a publicly available dataset relevant for engineers at Statoil ASA. The system provides different options, including visual, to formulate queries over ontologies and to display query answers. Optique 1.0 offers installation wizards that allow to extract ontologies from relational schemata, extract and define mappings connecting ontologies and schemata, and align and approximate ontologies. Moreover, the system offers highly optimised techniques for query answering.

1 Introduction

Accessing the *relevant* data in Big Data scenarios is increasingly difficult both for end-user and IT-experts, due to the *volume*, *variety*, *velocity*, and *complexity* dimensions of Big Data. This brings a high cost overhead in data access for large enterprises. For instance, in the oil and gas industry, engineers spend 30–70% of their time gathering and assessing the quality of data. The Optique project[1] [1, 2] advocates for a next generation of the well known *Ontology-Based Data Access* (OBDA) approach to address the data access problem. The project aims at solutions that reduce the cost of data access dramatically. In our demonstration we present the first version of the Optique system which we customised for the Norwegian Petroleum Directorate's (NPD) FactPages.[2]

OBDA systems address the data access problem by presenting a general ontology-based and end-user oriented query interface over heterogeneous data sources. The core elements in a classical OBDA systems are an *ontology*, describing

* The research was supported by the FP7 grant Optique (n. 318338).
** Corresponding author: `evgeny.kharlamov@cs.ox.ac.uk`
[1] `http://www.optique-project.eu/`
[2] `http://factpages.npd.no`

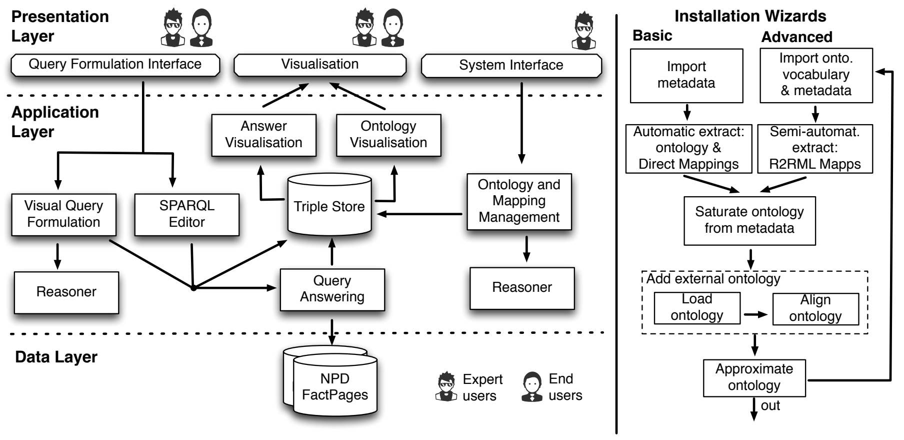


Fig. 1. *Left*: General architecture of the Optique 1.0 system; *Right*: installation process

the application domain, and a set of *mappings*, relating the ontological terms with the schemata of the underlying data sources. End-users formulate queries using the ontological terms and thus they are not required to understand the structure of the data sources. These queries are then automatically translated using the ontology and mappings into an executable code over the data sources.

State of the art OBDA systems, however, have shown among others the following limitations:

– The *usability* of OBDA systems is hampered by the need to use a formal query language. Even if the users know the ontological vocabulary, they may find difficult to formulate queries with several concepts and relationships.
– The *prerequisites* of OBDA, i.e., ontology and mappings, are in practice expensive to obtain. Additionally, they are not static artefacts and should evolve according to the new end-users' information requirements.
– The *efficiency* of the translation process and the execution of the queries is usually not sufficiently addressed in OBDA systems.

The first version of the Optique system, i.e., Optique 1.0, aims at partially overcoming the above limitations. Demonstration videos are available at following address: `http://www.cs.ox.ac.uk/isg/projects/Optique/demos/iswc2013/`.

2 System Overview

A general three-layer architecture of the Optique system is depicted in Figure 1 (*Left*). The current version of the system offers two main functionalities: to query/visualise data and install/maintain the ontology and the mappings. At the backend, the system also offers an efficient query processing mechanism.

Optique 1.0 allows to pose queries via a visual query formulation (VQF) interface, a SPARQL editor, or from a query catalog. VQF exploits reasoning in order to show both explicit and implicit domain knowledge to guide the formulation of the query.

Queries are executed by the Query Answering module based on Ontop system.[3] Ontop provides functionalities for rewriting SPARQL queries using the system's ontology and mappings, syntactic and semantic query optimisation, and query unfolding. Thus, high efficiency of query answering is guaranteed. Rewritten and unfolded queries are in SQL and they are executed over the NPD FactPages data, which is stored in a relational database. The query answers are converted into triples in order to confirm the format of the system's ontology, temporally stored in the system's triple store, and displayed to the user in a tabular way or on maps (using OpenStreetMap).

The installation and maintenance of the ontology and the mappings is done via the Ontology and Mapping Management component. Currently, this component includes two installation wizards: basic and advanced. In Figure 1 (*Right*) we depict workflows of the wizards. The basic wizard exploits the relational database metadata and automatically extracts an initial version of the ontology and direct mappings[4] to the ontology entities. The advanced wizard, unlike the basic one, requires the user intervention and an ontology vocabulary as input in order to (manually) create and edit R2RML mappings.[5] Both the basic and advanced wizards provide functionalities to align the bootstrapped ontology with a state of the art domain ontology and approximate the resulting ontology if it is outside the desired OWL 2 QL profile.[6] Alignment is performed using the ontology matching system LogMap,[7] which has shown to work well in practice and also includes mapping repair facilities.

Optique 1.0 is built on top of the Information Workbench[8] (IWB), a generic platform for semantic data management. The IWB provides a shared triple store for managing the assets of Optique 1.0, such as, ontologies, mappings, query logs, (excerpts of) query answers, database metadata, etc. The IWB also provides generic interfaces and APIs for semantic data management, e.g., ontology processing APIs. In addition to these backend data management capabilities, the IWB provides a flexible user interface which follows a semantic wiki approach, based on a rich, extensible pool of widgets for visualisation, interaction, mashup, and collaboration.

Finally, Optique 1.0 is customised for the NPD FactPages, which is a public, freely available dataset created to regulate and overlook the petroleum activities on the Norwegian Continental Shelf (NCS) and contains information collected from a wide range of activities on the NCS, e.g., operating companies, fields, discoveries, facilities, pipelines, and seismic surveys—both historic and current data. Its data has been converted and published as semantic web data [3], of which parts have been fed into the Optique 1.0 system.

[3] `http://ontop.inf.unibz.it/`
[4] `http://www.w3.org/TR/rdb-direct-mapping/`
[5] `http://www.w3.org/2001/sw/rdb2rdf/r2rml/`
[6] `http://www.w3.org/TR/owl2-profiles/`
[7] `http://code.google.com/p/logmap-matcher/`
[8] `http://www.fluidops.com/information-workbench/`

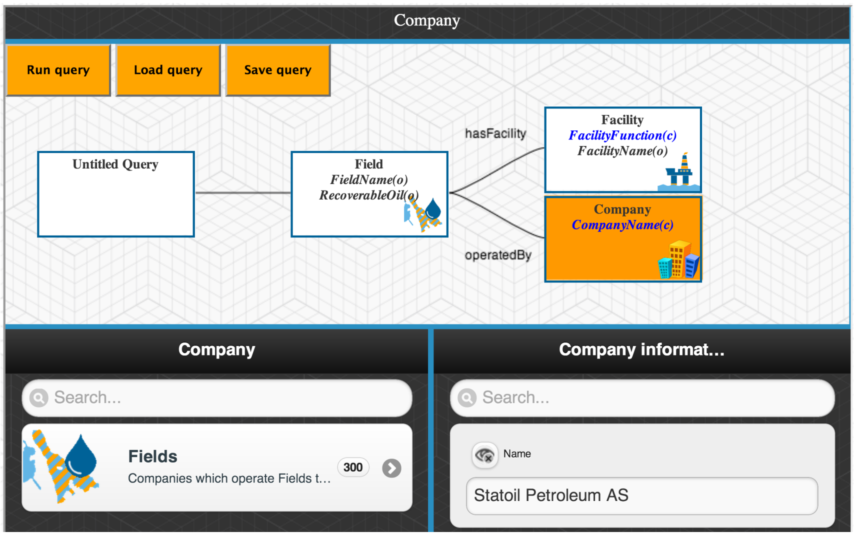


Fig. 2. Optique 1.0 System, visual query formulation component

3 Demonstration Details

During the demonstration we will describe the NPD FactPages and present functionalities of the Optique 1.0 system, with the focus on the following aspects: query formulation and execution, and system installation. These aspects will be illustrated on the NPD FactPages data. For the query formulation we will stress our visual query formulation tool that currently supports construction of tree-shaped conjunctive SPARQL queries. The demonstrated queries will be from the oil industry domain. An example query is: *"Find all fields that are operated by 'Statoil Petroleum AS' and which have a facility that produces oil"*; it can be seen in the screenshot of the VQF in Figure 2. We will run queries and present results both in tables and maps, e.g., the location of "Fields" and "Oil facilities" will be displayed on maps. Regarding the system's installation, we will present both basic and advanced wizards and guide through their steps, that is, loading metadata, extraction of an ontology and mappings, alignment with the domain ontology, and approximation of the integrated ontology. We will also show how to edit extracted direct mappings and define new R2RML mappings.

References

1. M. Giese et al. "Scalable End-user Access to Big Data". In: *Big Data Computing*. Ed. by R. Akerkar. Chapman and Hall/CRC, 2013.
2. E. Kharlamov et al. "Optique: Towards OBDA Systems for Industry". In: *ESWC postproceedings volume: Best Workshop Papers*. 2013.
3. M. G. Skjæveland, E. H. Lian, and I. Horrocks. "Publishing the Norwegian Petroleum Directorate's FactPages as Semantic Web Data". In: *The Semantic Web – ISWC 2013*. Ed. by H. Alani et al. Vol. 8219. LNCS. 2013.

Appendix L

Optique demo: 2014 (submitted)

This appendix reports the paper:

– E. Kharlamov, M. Giese, P. Haase, E. Jiménez-Ruiz, C. Pinkel, M. G. Skjæveland, A. Soylu, J. Trame, D. Zheleznyakov, C. Binnig, E. Bjorge, I Horrocks, and A. Waaler. Towards Ontology Based Data Access for Statoil. (Submitted).

Towards Ontology Based Data Access for Statoil

E. Kharlamov[1] M. Giese[2] P. Haase[3] E. Jiménez-Ruiz[1] C. Pinkel[3] M. G. Skjæveland[2]
A. Soylu[2] J. Trame[3] D. Zheleznyakov[1] C. Binnig[4] E. Bjørge[5] I. Horrocks[1] A. Waaler[2]
[1] University of Oxford; [2] University of Oslo; [3] fluid Operations AG; [4] DHBW Mannheim; [5] Statoil ASA

Abstract—**Ontology Based Data Access (OBDA) is a prominent approach to provide end users with high-level access to data via an ontology that is 'connected' to the data via mappings. State-of-the-art OBDA systems, however, suffer from limitations restricting their applicability in industry. Existing solutions focus on separate critical components of OBDA systems, while, to the best of our knowledge, there is no end-to-end OBDA solution allowing to deploy an OBDA system in an enterprise from scratch, effectively maintain and use it. In particular, development of necessary prerequisites to deploy an OBDA system, i.e., ontologies and mappings, as well as end user oriented query interfaces, are poorly addressed. The Optique platform provides an integrated end-to-end OBDA solution that addresses a number of practical challenges including the ones above. During the demonstration the attendees can try the platform with preconfigured scenarios from the petroleum industry and music domain, and try its end-to-end functionality: from deployment to query answering.**

I. Introduction

The growth of available information sources in enterprises requires new efficient methods for data access by domain experts whose ability to understand and analyse data is at the core of making business decisions. Currently in Statoil[1] and other data intensive companies there is a domination of centralised approaches, where an IT-expert translates information requests of domain experts into Extract-Transform-Load (ETL) processes to first integrate the data and then to apply predefined analytical reporting tools [8]. At the same time, in many scenarios an interactive data exploration, where domain experts want to access and analyse available data sources *directly*, without involving IT-experts, is of a high importance and centralised approaches are too heavy-weight and inflexible to do the job [7, 14]. In the Optique project [14] we aim at developing a direct data access solution that on the one hand would provide such access to Statoil's Exploration and Production Data Store (EPDS) and on the other hand would be generic enough to be applied in similar industries.

Challenges in providing domain experts with the direct data access include *(i)* the *complexity* of schemata that could contain hundreds and thousands of tables, e.g., EPDS has 3,000 tables with about 37,000 columns, and *(ii)* the *conceptual mismatch* between the language and structures that the domain experts use to describe the data, and the way the data is described and structured by database schema languages. Indeed, schemata are often integrated from autonomously evolving systems (yielding schema complexity), that have been adapted over years to the purpose of the applications they underly—EPDS was created 15 years ago–and not to the purposes of being intuitive for domain experts (yielding the conceptual mismatch). Further important practical challenges in providing direct data access are *(iii) formal query languages*, e.g., to access EPDS domain experts should be proficient in SQL, and *(iv) data incompleteness*. Regarding the latter challenge, considerable amount of Statoil exploration data are results of measurements taken during exploration activities by different teams and they are often incomplete and fragmented. Thus, a SQL encoding of a given information need over such data is inevitably complex, it may involve multiple data sources and tables referring to conceptually the same data stored in different places, e.g., queries over EPDS often contain thousands of words and have 50–200 joins.

Ontology Based Data Access (OBDA) [17] is a prominent, so-called *virtual*, approach to direct data access for end users, i.e., it provides an integration and access layer on top of databases while the data stays in the original stores. In OBDA users and data are mediated by an *ontology*, a semantically rich conceptual model,[2] and users formulate their information needs as queries over the ontology. These queries are enriched via *logical reasoning* over the ontology, translated into SQL over the underlying databases with the help of *mappings* (declarative specifications describing the relationship between the ontological vocabulary, e.g., 'wellbore' or 'oilfield', and the elements of the database schema) and finally executed over these databases automatically, without IT-experts intervention.

OBDA naturally addresses three out of four challenges above. Indeed, in contrast to a DB schema, an ontology describes a domain of interest rather than a structure of a DB and do it on a high level of abstraction in terms that are clear for domain experts, thus, avoiding the complexity of DB schemata required of Challenge (i), and the conceptual mismatch of Challenge (ii). Challenge (iv) can be addressed with the help of both mappings and ontologies. Indeed, an ontology is written in a logic based formal language e.g., W3C Web Ontology Language (OWL 2) as a set of OWL 2 axioms, and admits logical reasoning that allows to enrich ontological queries and address the incompleteness;[3] while mappings can relate one ontological term, e.g., 'wellbore', to many different parts of a DB.

State-of-the art OBDA systems, despite success stories in various scenarios, e.g. [5], have a number of important limitations. To the best of our knowledge there is no *end-to-end* OBDA solution providing both IT-experts and end users with the necessary tools to deploy an OBDA system in an enterprise from scratch, effectively maintain and use it. The existing solutions, e.g., [4, 5, 18, 19] focus on separate critical components of OBDA systems, and do not offer sufficient support for end users oriented query formulation discussed in Challenge (iii), as well as deployment and maintenance of OBDA systems, which is in practice expensive.

[1]The Norwegian oil and gas company, http://www.statoil.com/.

[2]Ontologies are common in many areas, e.g., medicine, Semantic Web [11].

[3]Many efficient off-the-shelf reasoning tools are available, e.g., [9, 19].

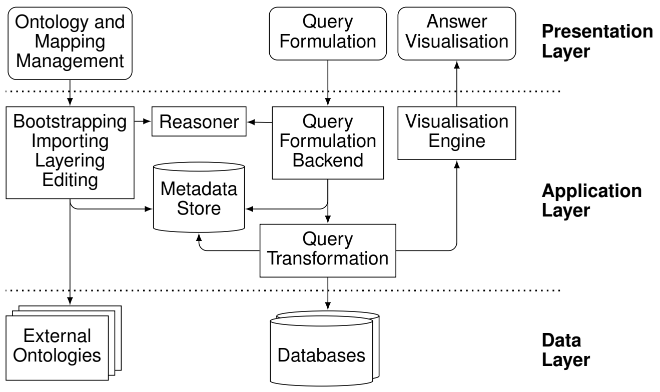


Fig. 1: General architecture of the Optique system

In the Optique project we have been developing an end-to-end OBDA system that satisfies Statoil requirements and addresses a number of practical challenges, including Challenges (i)-(iv) above. Our solution integrates in a unified platform a number of existing and novel components and allows one to deploy, maintain, and use the Optique platform in enterprises. Among the novel components, there is a deployment and maintenance module allowing to extract ontologies and mappings from relational databases in a semi-automatic fashion, integrate preexisting ontologies in an existing OBDA deployment instance, and edit mappings. We also developed a component for query formulation support that relies on novel techniques of projecting ontologies on graphs as well as components to visualise and browse query answers. We evaluated the platform with Statoil over EPDS and with other real world databases with encouraging results [22].

In this demonstration we show the platform's end-to-end functionality with the focus on its novel components. We demonstrate the platform on two preconfigured scenarios and allow the attendees to deploy it over either of the two databases underlying the scenarios, improve the deployment using the mapping editor, query the resulting deployment, see and browse query answers on maps, in tables, etc. Our first demo scenario is inspired by Statoil data:[4] we demonstrate the system on the NPD FactPages database [20], a publicly available data about petroleum activities on the Norwegian Continental Shelf that overlaps with EPDS. Since understanding NPD FactPages requires basic knowledge about the petroleum domain, we also demonstrate the platform over a large open music encyclopedia MusicBrainz [1]. The demo video illustrating the functionalities of our system and the demo system is available in [2].[5]

II. Overview of the Optique Platform

The general three-layer architecture of the Optique platform is illustrated in Figure 1. To deploy the platform over a relational DB, one can use its tools to extract ontologies and mappings from the DB, incorporate external ontologies, edit and author mappings of the resulting OBDA instance. After the system is deployed, the underlying DB can be queried using our visual query formulation tool that allows to compose queries by navigation over the system's ontology. Visually formulated queries are translated into SPARQL and sent to the query transformer for processing: query enrichment using the ontology and further unfolding with the mappings into SQL queries. We rely on the Ontop [19] query transformer, which is an integral component of the Optique platform. SQL queries are executed over the data sources underlying the system by the DBMSs of the sources. We offer a number of templates and widgets such as tables, timelines, maps, charts, etc., depending on the data modalities, to visualise and browse resulting query answers (see two screenshots of platform's answer visualisation in the bottom of Figure 3). The integration of the Optique platform is based on the Information Workbench [10], a generic and extensible platform for semantic data management, providing the platform with many base components, including interfaces and APIs as well as a triple store for managing ontologies, mappings, query logs, (excerpts of) query answers, DB metadata, etc.

[4]Due to privacy we cannot demo our solution on Statoil's corporate data.

[5]A very preliminary version of the Optique platform was presented in [13].

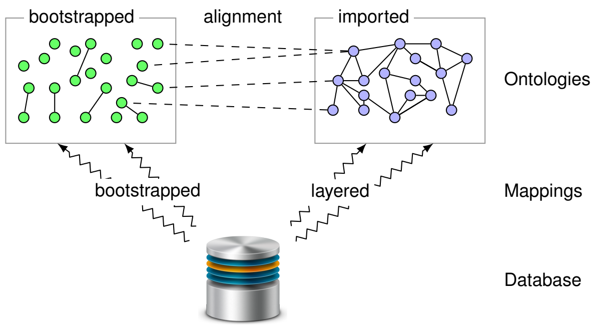


Fig. 2: Semi-automatic deployment approach

III. Deployment and Maintenance

The Optique platform provides semi-automatic support for deployment, which is schematically depicted in Figure 2. The platform supports different deployment scenarios. For example one can start with *bootstrapping*, i.e., a semi-automatic extraction of an ontology and mappings from the database. Then, one can *import* a pre-existing ontology and 'connect' it to the bootstrapped one via alignment with our LogMap ontology alignment system [12], which we have been developing during the last four years. This scenario can be applied, e.g., when the database schema or some of its fragments have a good correspondence with the domain of interest, or the available pre-existing ontology has a limited coverage of the domain of interest. Another possible scenario is to *layer* a pre-existing ontology directly over the database, i.e., to 'connect' it to the database schema with semi-automatically generated mappings. This scenario can address the case when there are several good ontologies available and they can serve as entry points to data for users with potentially different needs. The Optique platform supports ontologies expressible in the OWL 2 QL profile of OWL 2 ontology language, which was specifically designed for efficient data access. Imported or layered, OWL 2 ontologies that cannot be captured in OWL 2 QL are automatically approximated in OWL 2 QL using the technique of [6]. For mapping maintenance the platform offers a novel *mapping editor*. We now discuss Optique's modules in detail.

Mapping Bootstrapping Module automatically extracts so called *direct mappings* by relying on and extending the W3C specification, i.e., it extracts an ontological vocabulary from a relational schema, and it extracts mappings relating this vocabulary to the schema via SQL queries. The vocabulary consists of one class for each table that is not many-to-many, one property for each attribute and many-to-many table, and a special property for each foreign key (FK). The bootstrapped mappings are similar to view definitions, but serve a different purpose and technically more involved. In particular, mappings address a so-called *impedance mismatch* problem: ontologies are object oriented and objects are identified by URIs, while relational DBs contain tuples of values. We implemented several strategies to generate URIs for tuples that rely on heuristics as well as DB constraints, e.g., primary and foreign keys guarantee coherent URI generation for tuples from different tables. Figure 3 contains a screenshot of one of our bootstrapping wizards.

Ontology Bootstrapping Module enriches the bootstrapped vocabulary with OWL 2 axioms extracted from databases and implements a number of novel ontology bootstrapping techniques that are both schema (i.e., transforming explicit and implicit database constraints into ontological axioms) and data driven. For example, we turn FKs into OWL 2 axioms of domain and range restrictions on properties. Here we rely on FKs that are explicitly in schemas and candidate FKs that we derive by checking containment between attribute values in different tables. Computation of implicit FKs is motivated by our observation that in EPDS some FKs are not specified. We proposed several techniques to induce OWL 2 class and property hierarchies, as well as disjointness axioms over bootstrapped vocabularies. For example, by checking that all attribute names of a table T_1 occur in the attributes of a table T_2, we create a candidate OWL 2 class inclusion axiom saying that the class $C[T_1]$ corresponding to T_1 is a subclass of $C[T_2]$. We verify these candidate axioms by looking at the data instances and return a ranked list of axioms. By looking at the common set of attributes A between similar tables T_1 and T_2 (we have several notions of similarity), we induce a candidate class $C[A]$ corresponding to A and axioms that $C[T_1]$ and $C[T_2]$ are subclasses of $C[A]$; then, the user should assign a meaningful name to $C[A]$. By looking at tables T_1 and T_2 with similar structure while non-overlapping tuples we induce candidate disjointness axioms between $C[T_1]$ and $C[T_2]$. Each primary key that is not null or each unique attribute is represented as a functional property axiom. We developed a number of other techniques that we do not present here due to space limit. After the bootstrapper computes the set of all candidate axioms, it checks the set for logical consistency using the ontology reasoner HermiT [9], repairs the ontology if it is inconsistent, and presents the remaining candidate axioms to the users for verification, i.e., the user can edit, accept, or discard candidate axioms.

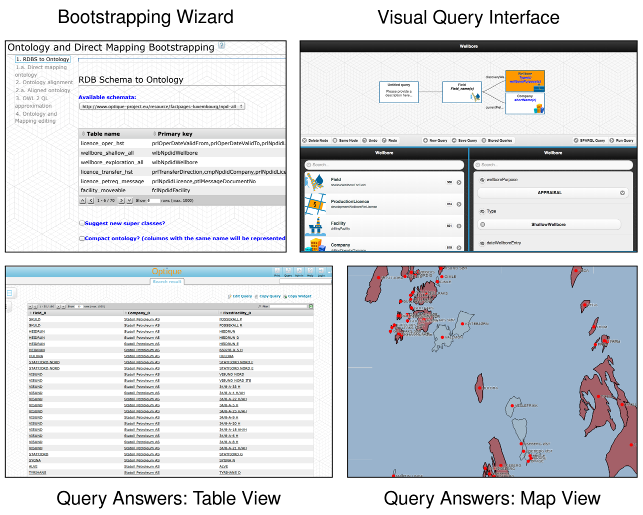
Ontology Importing Module allows to incorporate an existing ontology O_1 in the system by aligning it with the ontology O_2 already used by the system, e.g., with the bootstrapped one. Alignment introduces subclass and equivalence axioms between O_1's and O_2's classes and properties. Based on our experience with bootstrapping and importing for EPDS, we extended LogMap so that it guarantees that the resulting aligned ontology does not violate the so-called conservativity principle wrt the vocabulary of O_2, i.e., it does not add (potentially) undesired inclusions among O_2 classes [21].

Ontology Layering Module offers *layering* of an input ontology over an input DB schema resulting in a set of direct mappings between the ontology and the schema, by relying on the IncMap system [16] that we developed for the Optique platform. The module represents the ontology and schema as graphs 'preserving' their structure, computes ranked correspondences between elements of the graphs using lexical and structural similarities as in the Similarity Flooding algorithm of Melnik et al. converts the correspondences into direct mappings between the ontology and schema, and finally offer the mappings to the user for verification. Different to bootstrapping and importing, layering can map user-specified fragments of DB schemata to user-specified fragments of ontologies.

R2RML Mapping Editor of the Optique platform is tailored towards W3C R2RML mappings for which direct mappings is a special case, and was evaluated with encouraging results [15]. It provides an intuitive mapping visualisation, semi-automatic suggestions of mapping corrections, and step-by-step wizards for writing complex (non direct) mappings.

IV. Query Formulation

Visual query formulation component of the platform, OptiqueVQS [23], allows to compose conjunctive tree-shaped queries by navigation over ontologies, has a widget-based architecture, and exploits multiple representation and interaction paradigms for query composition. Ontologies are object oriented and data conforming to an ontology is a set of statements of the form 'wellbore(uri123)', 'locatedIn(uri123,uri456)', and 'oilfield(uri456)' that are instantiations of classes and properties with (URIs representing) objects. These data can be seen as a *data-graph* where nodes correspond to objects and labeled with classes, while edges correspond to properties and labeled with property names. Data-graphs are often enriched with extra nodes and edges to encode class and property hierarchies, thus, they can partially include information from ontological axioms. Furthermore, it is common to design query formulation interfaces over an ontology by visualising (relevant fragments of) its data-graph and the query formulation process boils down to navigation through the data-graph. For OBDA, where the ontological data is virtual and the user has access only to the ontological axioms, data-graph driven query interfaces are not appropriate. Thus, we developed novel techniques to 'project' axioms rather than data in a graph structure as an *axiom-graph*: an OWL 2 axiom is projected into a set of nodes and edges relating them, where nodes correspond to classes and edges to properties [3]. Projecting axioms to graphs is not a trivial task since axioms are first-order logic formulae and do not have an immediate correspondence to graphs. In OptiqueVQS query construction is iterative, i.e., users construct queries step-by-step, and it boils down to navigation over an axiom-graph. At the moment the system supports axiom-graphs encoding those types of axioms which can be bootstrapped by the deployment module, including class hierarchies, and domain and range restrictions. E.g., consider two axioms saying that the classes 'wellbore' and 'oilfield' are respectively a domain and range of a property 'locatedIn', then the corresponding axiom-graph contains two nodes labeled respectively with 'wellbore' and 'oilfield', and one edge connecting these nodes labeled with 'locatedIn'. In Figure 3 there is a screenshot of OptiqueVQS, where in the upper part there is a query constructed by the user and in the lower-left part there is a fragment of axiom-graph relevant to the constructed query. Important feature of OptiqueVQS is that it does not require to store the axiom-graph: during each query construction session we compute (using logical reasoning with



Ontology and Direct Mapping Bootstrapping

1. RDBS to Ontology
1.a. Direct mapping ontology
2. Ontology alignment
2.a. Aligned ontology
3. OWL 2 QL approximation
4. Ontology and Mapping editing

RDB Schema to Ontology

Available schemata: http://www.optique-project.eu/resource/factpages-luxembourg/npd-all

Table name	Primary key
licence_oper_hst	prlOperDateValidFrom,prlOperDateValidTo,prlNpdidL
wellbore_shallow_all	wlbNpdidWellbore
wellbore_exploration_all	wlbNpdidWellbore
licence_transfer_hst	prlTransferDirection,cmpNpdidCompany,prlNpdidLice
licence_petreg_message	prlNpdidLicence,ptlMessageDocumentNo
facility_moveable	fclNpdidFacility

1 - 6 / 70 Show 6 rows (max. 1000)

Suggest new super classes?

Compact ontology? (columns with the same name will be represented

Fig. 3: Screenshots of the Optique platform

HermiT) relevant fragments of axiom-graph on-the-fly and present to the user. As we observed in our user studies [22], a purely axiom driven query interface suffers from important practical limitations, e.g., it does not allow users to set specific data values in queries, e.g., company names. To address this issue we enrich axiom-graphs with data annotations which we precompute, i.e., materialise, from the DBs underlying a given OBDA deployment instance by 'executing' relevant mappings. E.g., for EPDS we precomputed values that are frequently used, rarely changed, and from relatively small domains; this includes names of companies and oilfield, geolocations, ranges of numerical values, e.g., min/max possible depth of wellbores. Data values are visualised using sliders, drop boxes, etc., see the lower-right part of the interface in Figure 3.

V. Demonstration Scenario

We will demonstrate the Optique platform end-to-end, i.e., from deployment to query answering over two databases. Moreover, for these databases we prepared OBDA deployments with fine tuned ontologies and mappings and the attendees of the demo will be able to formulate queries over these deployments, load queries from query catalogs, execute queries and browse query answers on maps and in tables. We next describe the demonstration scenarios in detail.

Demonstration on NPD FactPages One deployment of the Optique platform is made over the NPD FactPages [20], an important public dataset heavily used in the oil and gas industry. This DB has 70 tables, 276 different attributes, 96 foreign keys, and about 50 MB of mostly aggregated data, e.g., seismic surveys. The choice of this demo database was motivated by its importance for the oil and gas industry and our work with Statoil within the Optique project. This deployment was tested by Statoil engineers who gave us positive feedback. Usage of this deployment requires a basic knowledge of geophysics.

Demonstration on MusicBrainz Database The other deployment of the Optique platform is made over the MusicBrainz database [1], which is an open music encyclopaedia that contains music information about roughly 830,000 artists, 1.2 million releases, and 13.2 million recordings. This domain does not require any special knowledge, so it is easy for anyone to use. This deployment was tested by students and the results were encouraging, e.g., students were able to accomplish query formulation tasks using OptiqueVQS.

End-to-End Demonstration Besides querying the two pre-configured deployments, the demo attendees will be able to deploy the Optique platform over both NPD FactPages and MusicBrainz databases and then query their own deployments. Specifically, one will be able to bootstrap an ontology and mappings from these databases, either *(i)* in a 'simple' mode, suitable for inexperienced users, with bootstrapping performed in an automatic regime using default parameters, or *(ii)* in a step-by-step mode that allows a user to tune the deployment parameters, e.g., to discover implicit database constraints and propagate them to the bootstrapped ontology. Then, one will be able to import pre-existing ontologies from our ontology catalogue. This can be performed in two ways: *(i)* either after the bootstrapping, in which case the imported and bootstrapped ontologies will be aligned, or *(ii)* using our ontology layering component, thus, skipping the bootstrapping step. Moreover, the attendees of the demo will be able to manually edit mappings with our mapping editor. Finally, they will be able to query the resulting deployments and browse query answers.

VI. References

[1] URL: http://musicbrainz.org/statistics.
[2] URL: http://fact-pages.fluidops.net/resource/demoICDE.
[3] M. Arenas et al. Faceted Search over Ontology-Enhanced RDF Data. In: *CIKM*. 2014.
[4] C. Bizer and A. Seaborne. D2RQ—treating non-RDF databases as virtual RDF graphs. In: *ISWC*. 2004.
[5] D. Calvanese et al. The MASTRO System for Ontology-Based Data Access. In: *Semantic Web* 2.1 (2011).
[6] M. Console et al. Efficient Approximation in DL-Lite of OWL 2 Ontologies. In: *DL*. 2013.
[7] J. Crompton. *Keynote talk at the W3C Workshop on Sem. Web in Oil & Gas Industry*. http://www.w3.org /2008/12/ogws-slides/Crompton.pdf. 2008.
[8] A. Doan et al. *Principles of Data Integration*. Morg. Kauf.'12.
[9] Glimm et al. Optimising Ontology Classification. In: *ISWC'10*.
[10] P. Haase et al. The Information Workbench as a Self-Service Platform for Linked Data Applications. In: *WWW*. 2012.
[11] I. Horrocks. What are ontologies good for? In: *Evolution of Semantic Systems*. Springer, 2013.
[12] E. Jimenez-Ruiz et al. Large-Scale Interactive Ontology Matching: Algorithms and Implementation. In: *ECAI'12*.
[13] E. Kharlamov et al. Optique 1.0: Semantic Access to Big Data: The Case of Norwegian Petroleum Directorate's FactPages. In: *ISWC (Posters & Demos)*. 2013.
[14] E. Kharlamov et al. Optique: Towards OBDA Systems for Industry. In: *ESWC (SE)*. 2013.
[15] C. Pinkel et al. How to Best Find a Partner? An Evaluation of Editing Approaches to Construct R2RML Mappings. In: *ESWC*. 2014.
[16] C. Pinkel et al. IncMap: Pay as You Go Matching of Relational Schemata to OWL Ontologies. In: *OM*. 2013.
[17] A. Poggi et al. Linking Data to Ontologies. In: *J. Data Semantics* 10 (2008).
[18] F. Priyatna et al. Formalisation and Experiences of R2RML-based SPARQL to SQL query translation using Morph. In: *WWW*. 2014.
[19] M. Rodriguez-Muro et al. Onto-logy-Based Data Access: Ontop of Databases. In: *ISWC*. 2013.
[20] M. G. Skjæveland et al. Publishing the NPD FactPages as Semantic Web Data. In: *ISWC*. 2013.
[21] A. Solimando et al. Detecting and Correcting Conservativity Principle Violations in Onto-to-Onto Mappings. In: *ISWC'14*.
[22] A. Soylu et al. Experiencing OptiqueVQS: A Multi-paradigm and Ontology-based Visual Query System for End Users. In: *Under Review*.
[23] A. Soylu et al. OptiqueVQS: Towards an Ontology-Based Visual Query System for Big Data. In: *MEDES*. 2013.

Appendix M

ISWC 2014: Ontology Alignment for Query Answering

This appendix reports the paper:

- Alessandro Solimando, Ernesto Jimenez-Ruiz, Christoph Pinkel. Evaluating Ontology Alignment Systems in Query Answering Tasks. In Proceedings of the International Semantic Web Conference 2014 (Poster paper)

Evaluating Ontology Alignment Systems in Query Answering Tasks

Alessandro Solimando[1], Ernesto Jiménez-Ruiz[2], and Christoph Pinkel[3]

[1] Dipartimento di Informatica, Bioingegneria, Robotica e Ingegneria dei Sistemi, Università di Genova, Italy
[2] Department of Computer Science, University of Oxford, UK
[3] fluid Operations AG, Walldorf, Germany

Abstract. Ontology matching receives increasing attention and gained importance in more recent applications such as ontology-based data access (OBDA). However, query answering over aligned ontologies has not been addressed by any evaluation initiative so far. A novel Ontology Alignment Evaluation Initiative (OAEI) track, Ontology Alignment for Query Answering (OA4QA), introduced in the 2014 evaluation campaign, aims at bridging this gap in the practical evaluation of matching systems w.r.t. this key usage.

1 Introduction

Ontologies play a key role in the development of the Semantic Web and are being used in many application domains such as biomedicine and energy industry. An application domain may have been modeled with different points of view and purposes. This situation usually leads to the development of different ontologies that intuitively overlap, but they use different naming and modeling conventions.

The problem of (semi-)automatically computing mappings between independently developed ontologies is usually referred to as the *ontology matching problem*. A number of sophisticated ontology matching systems have been developed in the last years [5]. Ontology matching systems, however, rely on lexical and structural heuristics and the integration of the input ontologies and the mappings may lead to many undesired logical consequences. In [1] three principles were proposed to minimize the number of potentially unintended consequences, namely: *(i) consistency principle*, the mappings should not lead to unsatisfiable classes in the integrated ontology; *(ii) locality principle*, the mappings should link entities that have similar *neighbourhoods*; *(iii) conservativity principle*, the mappings should not introduce alterations in the classification of the input ontologies. The occurrence of these violations is frequent, even in the reference mapping sets of the Ontology Alignment Evaluation Initiative[4] (*OAEI*) [6].

Violations to these principles may hinder the usefulness of ontology mappings. The practical effect of these violations, however, is clearly evident when ontology alignments are involved in complex tasks such as query answering [4].

[4] `http://oaei.ontologymatching.org/`

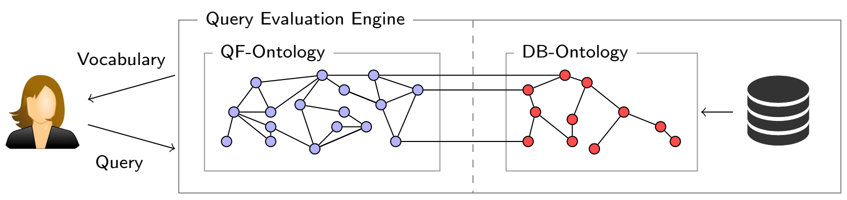


Fig. 1. Ontology Alignment in an OBDA Scenario

The traditional tracks of *OAEI* evaluate ontology matching systems w.r.t. scalability, multi-lingual support, instance matching, reuse of background knowledge, etc. Systems' effectiveness is, however, only assessed by means of classical information retrieval metrics (*i.e.*, precision, recall and f-measure) w.r.t. a manually-curated reference alignment, provided by the organisers. The new *OA4QA* track[5] evaluates those same metrics, but w.r.t. the ability of the generated alignments to enable the answer of a set of queries in an OBDA scenario, where several ontologies exist. Figure 1 shows an OBDA scenario where the first ontology provides the vocabulary to formulate the queries (QF-Ontology) and the second is linked to the data and it is not visible to the users (DB-Ontology). Such OBDA scenario is presented in real-world use cases (*e.g.*, Optique project[6] [2, 6]). The integration via ontology alignment is required since only the vocabulary of the DB-Ontology is connected to the data. The *OA4QA* will also be key for investigating the effects of logical violations affecting the computed alignments, and evaluating the effectiveness of the repair strategies employed by the matchers.

2 Ontology Alignment for Query Answering

This section describes the considered dataset and its extensions (Section 2.1), the query processing engine (Section 2.2), and the evaluation metrics (Section 2.3).

2.1 Dataset

The set of ontologies coincides with that of the *conference* track,[7] in order to facilitate the understanding of the queries and query results. The dataset is however extended with synthetic ABoxes, extracted from the *DBLP* dataset.[8]

Given a query q expressed using the vocabulary of ontology $\mathcal{O}_1$, another ontology $\mathcal{O}_2$ enriched with syntethic data is chosen. Finally, the query is executed over the aligned ontology $\mathcal{O}_1 \cup \mathcal{M} \cup \mathcal{O}_2$, where $\mathcal{M}$ is an alignment between $\mathcal{O}_1$ and $\mathcal{O}_2$. Referring to Figure 1, $\mathcal{O}_1$ plays the role of QF-Ontology, while $\mathcal{O}_2$ that of DB-Ontology.

[5] http://www.cs.ox.ac.uk/isg/projects/Optique/oaei/oa4qa/
[6] http://www.optique-project.eu/
[7] http://oaei.ontologymatching.org/2014/conference/index.html
[8] http://dblp.uni-trier.de/xml/

2.2 Query Evaluation Engine

The evaluation engine considered is an extension of the OWL 2 reasoner *HermiT*, known as *OWL-BGP*[9] [3]. OWL-BGP is able to process SPARQL queries in the SPARQL-OWL fragment, under the *OWL 2 Direct Semantics entailment regime*.[10] The queries employed in the *OA4QA* track are standard conjunctive queries, that are fully supported by the more expressive SPARQL-OWL fragment. SPARQL-OWL, for instance, also support queries where variables occur within complex class expressions or bind to class or property names.

2.3 Evaluation Metrics and Gold Standard

As already discussed in Section 1, the evaluation metrics used for the *OA4QA* track are the classic information retrieval ones (*i.e.*, precision, recall and f-measure), but on the result set of the query evaluation. In order to compute the gold standard for query results, the publicly available reference alignments *ra1* has been manually revised. The aforementioned metrics are then evaluated, for each alignment computed by the different matching tools, against the *ra1*, and manually repaired version of *ra1* from conservativity and consistency violations.

Three categories of queries will be considered in *OA4QA*: *(i)* basic, *(ii)* queries involving violations, *(iii)* advanced queries involving nontrivial mappings.

2.4 Impact of the Mappings in the Query Results

As an illustrative example, consider the aligned ontology $\mathcal{O}_{\mathcal{U}}$ computed using *confof* and *ekaw* as input ontologies ($\mathcal{O}_{confof}$ and $\mathcal{O}_{ekaw}$, respectively), and the *ra1* reference alignment between them. $\mathcal{O}_{\mathcal{U}}$ entails $ekaw{:}Student \sqsubseteq ekaw{:}Conf\_Participant$, while O_{ekaw} does not, and therefore this represents a conservativity principle violation. Clearly, the result set for the query $q(x) \leftarrow ekaw{:}Conf\_Participant(x)$ will erroneously contain any student not actually participating at the conference. The explanation for this entailment in $\mathcal{O}_{\mathcal{U}}$ is given below, where Axioms 1 and 3 are mappings from the reference alignment.

$$confof{:}Scholar \equiv ekaw{:}Student \tag{1}$$

$$confof{:}Scholar \sqsubseteq confof{:}Participant \tag{2}$$

$$confof{:}Participant \equiv ekaw{:}Conf\_Participant \tag{3}$$

The softening of Axiom 3 into $confof{:}Participant \sqsupseteq ekaw{:}Conf\_Participant$ represents a possible repair for the aforementioned violation.

3 Preliminary Evaluation

In Table 1 [11] a preliminary evaluation using the alignments of the *OAEI 2013* participants and the following queries is shown: *(i)* $q_1(x) \leftarrow ekaw{:}Author(x)$,

[9] `https://code.google.com/p/owl-bgp/`

[10] `http://www.w3.org/TR/2010/WD-sparql11-entailment-20100126/#id45013`

[11] $\#q(x)$ refers to the cardinality of the result set.

Category	**Query**	$\#\mathcal{M}$	**Reference Alignment**				**Repaired Alignment**			
			#q(x)	**Prec.**	**Rec.**	**F-meas.**	**#q(x)**	**Prec.**	**Rec.**	**F-meas.**
Basic	q_1	5	98	1	1	1	98	1	1	1
Violations	q_2	4	53	0.8	1	0.83	38	0.57	1	0.68
Advanced	q_3	7	-	-	-	-	182	1	0.5	0.67

Table 1. Preliminary query answering results for the OAEI 2013 alignments

over the ontology pair $\langle cmt, ekaw\rangle$; *(ii)* $q_2(x) \leftarrow ekaw{:}Conf\_Participant(x)$, over $\langle confof, ekaw\rangle$, involving the violation described in Section 2.4; *(iii)* and $q_3(x) \leftarrow confof{:}Reception(x) \cup confof{:}Banquet(x) \cup confof{:}Trip(x)$, over $\langle confof, edas\rangle$. The evaluation[12] shows the negative effect on precision of logical flaws affecting the computed alignments (q_2) and a lowering in recall due to missing mapping (q_3). For q_3 the results w.r.t. the reference alignment (*ra1*) are missing due to the unsatisfiability of the aligned ontology $\mathcal{O}_{confof} \cup \mathcal{O}_{edas} \cup ra1$.

4 Conclusions and Future Work

We have presented the novel OAEI track addressing query answering over pairs of ontologies aligned by a set of ontology-to-ontology mappings. From the preliminary evaluation the main limits of the traditional evaluation, for what concerns logical violations of the alignments, clearly emerged. As a future work we plan to cover increasingly complex queries and ontologies, including the ones in the Optique use case [6]. We also plan to consider more complex scenarios involving a single QF-Ontology aligned with several DB-Ontologies.

Acknowledgements. This work was supported by the EU FP7 IP project Optique (no. 318338), the MIUR project CINA (Compositionality, Interaction, Negotiation, Autonomicity for the future ICT society) and the EPSRC project Score!.

References

1. Jiménez-Ruiz, E., Cuenca Grau, B., Horrocks, I., Berlanga, R.: Logic-based Assessment of the Compatibility of UMLS Ontology Sources. J. Biomed. Semant. (2011)
2. Kharlamov, E., et al.: Optique 1.0: Semantic Access to Big Data: The Case of Norwegian Petroleum Directorate's FactPages. ISWC (Posters & Demos) (2013)
3. Kollia, I., Glimm, B., Horrocks, I.: SPARQL query answering over OWL ontologies. In: The Semantic Web: Research and Applications, pp. 382–396. Springer (2011)
4. Meilicke, C.: Alignments Incoherency in Ontology Matching. Ph.D. thesis, University of Mannheim (2011)
5. Shvaiko, P., Euzenat, J.: Ontology Matching: State of the Art and Future Challenges. IEEE Transactions on Knowl. and Data Eng. (TKDE) (2012)
6. Solimando, A., Jiménez-Ruiz, E., Guerrini, G.: Detecting and Correcting Conservativity Principle Violations in Ontology-to-Ontology Mappings. In: International Semantic Web Conference (2014)

[12] Out of the 26 alignments of OAEI 2013, only the ones shown in column $\#\mathcal{M}$ were able to produce a result (either for logical problems or for an empty result set due to missing mappings). Reported precision/recall values are averaged values.