Semantic web semantics
(M2R SWXO lecture notes)

Jérôme Euzenat
INRIA & Univ. Grenoble Alpes,
Laboratoire d'informatique de Grenoble,
Montbonnot, France
Jerome.Euzenat@inria.fr

April 28, 2017

Chapter 1

Introduction

The web has been constantly evolving from a distributed hypertext system to a very large information processing machine. As fast as it is, this evolution is grounded on theoretical principles borrowing to several fields of computer science such as programming languages, data bases, structured documentation, logic and artificial intelligence. The smooth operation of the past and future web at a large scale is relying on these foundations.

These lecture notes introduce the semantics of knowledge representation on the web (semantic web technologies). Further, they aim at acquainting students with the techniques of knowledge representation so that they can use them in other contexts. In particular, we aim at providing:

- a deep understanding of these technologies, at least deeper than the simple language approach;
- a concrete application example of logics, out of logics itself.

The semantic web extends the web with richer and more precise information because it is expressed in a formal language using a vocabulary defined in an ontology (a structured vocabulary of concepts and properties defined in a logic). Ontologies are used for describing Web resource content and reasoning about these resources formally. We introduce the semantic Web languages (RDF, RDFS, OWL) and show their relations with knowledge representation formalisms (conceptual graphs, description logics) and XML. This provides the tools for reasoning with ontologies and, in particular, to evaluate queries. However, the distributed nature of the web leads to heterogeneous ontologies which must be matched before using them. We show how to match ontologies and how to semantically interpret the relations between ontologies. Finally, this is applied to network of peers using knowledge together.

The material is presented so as to be as generic as possible, however it is illustrated through specific and practical semantic web technologies (mostly language).

1.1 Historical notes on the semantic web

The idea of using knowledge representation on the worldwide web appeared rapidly after the development of the web. Systems like SHOE [LUKE et al. 1997] and Ontobroker [FENSEL, DECKER, et al. 1998] integrated formal knowledge representation in web pages while Ontoserver [FARQUHAR et al. 1995] and HyTroeps [EUZENAT 1996] used the web to browse and edit knowledge.

In 1998, Tim Berners-Lee described the principles of what he called the 'semantic web'. He already had launched, within the W3C[1], the development of the RDF language whose initial goal was to annotate web pages. At that time, he wrote '*A Semantic Web is not Artificial Intelligence*', but later he added that it was '*Knowledge Representation goes Global*' [BERNERS-LEE 1998].

[1] *WorldWide Web Consortium*: the organisation in charge of recommending web technologies.

In 2000, this idea had not taken off nor been subject to ambitious developments. But the US DARPA launched the DAML (*DARPA Agent Markup Language*) program directed by James Hendler, the initiator of SHOE. Right afterwards, was held a seminar on '*Semantics for the Web*' [FENSEL, HENDLER, et al. 2003] which has steered the EU OntoWeb thematic network, itself leading to the Knowledge Web network of excellence. Such efforts have involved many artificial intelligence researchers in the development of semantic web technologies [EUZENAT (ED.) 2002].

The semantic web is often presented as a 'web for machines': it aims at representing knowledge on the web with formal languages that can be processed by machines. For that purpose, it is natural to use techniques developed in artificial intelligence.

Indeed, the main target application for the semantic web is information retrieval: being able to retrieve precise information on the web because documents are annotated with 'semantic metadata'. Although this goal seemed remote in 2000, it is nowadays far more tangible. In the interval, we witnessed the use of of Open Graph by Facebook, the definition of the lightweight ontology schema.org by four of the main search engines (Bing, Google, Yahoo, Yandex), and the use by Google of the *knowledge graph* providing structured answers to queries instead of lists of documents.

Many other types of applications have emerged, each based on the same basis: the annotation of resources using the same knowledge representation language. Such applications are referred to as applications of semantic technologies. In particular:

- Semantic web services in which service web interfaces (input/output) are semantically annotated,
- Semantic peer-to-peer systems in which shared resources are semantically annotated,
- Semantic social networks in which social relations and people profiles are semantically annotated,
- Semantic desktop in which personal informations (agenda, address book, etc.) are semantically expressed,
- Semantic web of things in which sensors, effectors and the information they exchange are semantically annotated,
- The web of data in which data sources are expressed and linked in semantic web languages.

The semantic web has thus spread out of its initial playground. It is now a broad experimentation field in which are investigated various problems such as trust in peer-to-peer systems, statistical data integration, or smart city monitoring. As web technologies, semantic technologies have the potential to progressively affect all computer developments [JANOWICZ et al. 2015].

1.2 Lecture note format and topic dependencies

This document constitutes the lecture notes of a master course (see §1.3) that has been taught entirely or partially between 2007 and 2017 in various formats at the University of Grenoble following my previous lectures on Semantics of knowledge representation [EUZENAT 2002] (in French).

In its entirety, the course is made of 7 sessions of 3h.

3h00 Expressing information (RDF)
6h00 Modelling knowledge (RDFS and OWL)
3h00 Querying web data (SPARQL)
3h00 SPARQL Extensions
3h00 Ontology alignments
3h00 Networks of ontologies

It is further divided in three parts: Expressing knowledge (data and ontologies), Querying and Networks of ontologies.

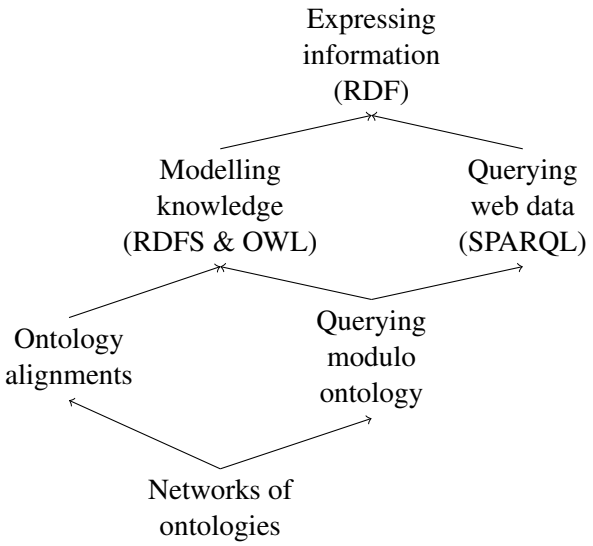


Figure 1.1: Dependencies between the various topics of these lectures.

The dependencies between the main parts are provided in Figure 1.1. However, the course may also be given or read in a reduced form by considering only Chapter 2, 3, 4, 5, 8, 8.2, 10, and 12.1.

The evolution of this course is the imperfect result of a struggle between antagonist forces towards simplification, e.g., mostly ignoring data types, and complexification, e.g., adding features of OWL 2 and SPARQL 1.1.

1.3 Context in the semantic web course

This section provides a connection with the part of the course dedicated to the foundations of XML taught by Pierre Genevès and Nabil Layaïda.

They have considered expressing information within the XML language. XML is not a very expressive language: it provides a structure which stores information and this information has to be extracted in order to be accessed and exploited. It does not allow for expressing knowledge in terms of descriptions which will be used for guiding this interpretation.

XML provides a database point of view about knowledge representation in which the power of what can be done with the language depends on the power of the query language (Xpath, Xquery, etc.) and the representation is a store.

In the context of the semantic web, this language is provided with a semantics specifying how to interpret the structure. This allows for defining general statements which will apply to the relevant data whatever its syntactic form.

Doing so requires the definition of a logic, specifying the semantics of statements (or formulas) and identifying the possible models of a document or a set of documents. Answering queries will then have to be performed through inference sanctioned by their correctness and completeness with regard to the semantics.

There is anyway a continuity between these manipulation of the data on the web. It starts with extracting data from documents through navigation:

$$\text{Document (XML)} \vdash \text{Query(xpath)}$$

The information can also be extracted modulo a document schema. Although schemas are here in the first place for specifying acceptable document structures, they often provide shortcuts and can help producing more guarantees on query answering:

$$\text{Document (XML) + Schema (DTD, XML Schema)} \vdash \text{Query (xpath)}$$

In addition, sophisticated query languages provide the possibility to reason about queries through their algebraic structures and the opportunity to arrange extracted information in a better way:

$$\text{Document (XML) + Schema (XML Schema)} \vdash \text{Query (xquery)}$$

Semantic web languages, as presented here, manipulate data roughly in the same way as this is done in XML, hence:

$$\text{Data (RDF)} \models \phi \text{ (RDF)}$$

however, we use $\models$ to emphasise that the operations for extracting information are defined semantically. Similarly to XQuery, the SPARQL query language has been defined for querying RDF data:

$$\text{Data (RDF)} \vdash \text{Query (SPARQL)}$$

Although, defined syntactically in the SPARQL recommendation, it is preferable to define SPARQL semantically:

$$\text{Data (RDF)} \models \text{Query (SPARQL)}$$

and to show the completeness of the syntactic description with respect to the RDF semantics. In addition, semantic web technologies allow for expressing ontologies which are theories constraining how data must be interpreted. RDF and its extension RDFS (RDF Schema) together with OWL form the three formal logics recommended by W3C for representing knowledge on the semantic web. In this presentation, the only operation which will be considered is the test that data modulo some ontologies entails a particular formula:

$$\text{Data (RDF) + Ontology (RDF Schema, OWL)} \models \phi \text{ (RDF)}$$

We will concentrate on the entailment test and consider further elaborate query languages later. Finally, the semantic web inherits the distributed aspect of the web. This means that independent data and ontology sources provide information which has to be interpreted together. For that purpose, one may consider a set Ω of ontologies and data sources (still described in OWL and RDF) and a set of alignments Λ which expresses relations between entities in the ontologies. Hence, extracting information from the semantic web, is obtained within such a network of ontologies, through:

$$\text{Data (RDF) + Ontology (RDF Schema, OWL)} \models_{\Omega,\Lambda} \phi \text{ (RDF)}$$

The frontiers between the "syntactic web" and the semantic web tends to vanish as XML gain expressiveness (with XML Schema and Xquery) and RDF allows for minimal reasoning. The definition of XML and query languages, such as XPath, in logics, such as the μ-calculus, calls for a closer understanding of both worlds. This is one of the main reasons of putting these lectures together. Developing semantic web applications, like in many other cases, is characterised by a trade-off between the expressivity of the languages used and the complexity of reasoning.

So, expressing knowledge and information on the web depends on different languages:

- data language (XML, RDF);
- ontology language (XML Schema, RDF Schema, OWL)
- query language (Xpath, Xquery, SPARQL)
- distributed aspect (Alignments)

each one needing to be semantically defined.

1.4 Exercises

We offer exercises which are usually based on former exams. They have been dispatched at the end of chapters so as to be considered as soon as the chapter is covered. The exams (90mn) were respectively made of:

- Exercises 1, 8, 18 (and additionally 28);
- Exercises 2, 9, 10, 20, and 22;
- Exercises 3, 11, 21, 23, and 27.
- Exercises 12, 13, and 24.
- Exercises 4, 14, and 25.
- Exercises 5, 15, and 26.

and those of 60mn made of:

- Exercises 6 and 16
- Exercises 7, 17 and 19

1.5 Further information

The web site of the course is `http://exmo.inria.fr/teaching/sw/`. It contains references to other material as well as information on the first part of the course.

Any comment, issue or clarification enquiry may be send by email to the author:

Jerome.Euzenat@inria.fr

Further books covering parts of these topics are [ANTONIOU and VAN HARMELEN 2008; HITZLER et al. 2009].

1.6 Acknowledgements

Most of the material presented here comes from the collective effort for establishing W3C recommendations.

Other parts of these notes have been extracted from part of my own works some of them non published. Finally, important parts of Chapter 5 and 6.2 come from my student Faisal Alkhateeb's thesis [ALKHATEEB 2008] and parts of Chapter 10 and 11 are results from Antoine Zimmermann's PhD dissertation [ZIMMERMANN 2008]. Some pieces of text and particularly interludes have been written for a book chapter with Marie-Christine Rousset [EUZENAT and ROUSSET 2017].

Thanks to the students of these courses for their questions and criticisms.

Contents

1 Introduction **3**
- 1.1 Historical notes on the semantic web . . . 3
- 1.2 Lecture note format and topic dependencies . . . 4
- 1.3 Context in the semantic web course . . . 5
- 1.4 Exercises . . . 7
- 1.5 Further information . . . 7
- 1.6 Acknowledgements . . . 7

I Graphs and ontologies **13**

2 RDF: Resource description framework **17**
- 2.1 IRI (and XML): the basic layer . . . 17
- 2.2 RDF Syntax . . . 17
- 2.3 Simple RDF Semantics . . . 20
- 2.4 Inference mechanism . . . 24
- 2.5 Computational properties . . . 25
- 2.6 Conclusion . . . 25
- 2.7 Exercises . . . 26

Interlude 1: The web of data **31**

3 Ontology languages **33**
- 3.1 RDF Schema . . . 33
- 3.2 The web ontology language OWL . . . 41
- 3.3 Conclusion . . . 53
- 3.4 Exercises . . . 53

Interlude 2: Rules **59**

II Queries **61**

4 Querying the semantic web **63**
- 4.1 Motivation . . . 63
- 4.2 What is a query language? . . . 63

5 Querying RDF with SPARQL **65**
5.1 Syntax . . . 65
5.2 SPARQL Semantics . . . 68
5.3 Algebraic manipulation . . . 71
5.4 Entailment regimes . . . 72
5.5 Conclusion . . . 73
5.6 Exercises . . . 73

6 Extending SPARQL **75**
6.1 NSPARQL . . . 75
6.2 PSPARQL . . . 78
6.3 cpSPARQL and CPSPARQL . . . 80

Interlude 3: Streams and navigation **87**

7 Querying modulo ontologies **89**
7.1 RDF(S) closure and query answering . . . 90
7.2 Ontologies expansion as path navigation . . . 91
7.3 Querying modulo DL-lite . . . 96
7.4 Conclusion . . . 96
7.5 Exercises . . . 96

III Networks of ontologies **99**

8 Alignments and networks of ontologies **101**
8.1 Motivation . . . 101
8.2 Alignment syntax . . . 103
8.3 Networks of ontologies . . . 108
8.4 Conclusion . . . 111

9 Alignment algebra **113**
9.1 Algebras of relations . . . 114
9.2 Aggregating matcher results . . . 116
9.3 Composing alignments . . . 117
9.4 Algebraic reasoning with alignments . . . 118
9.5 Algebra granularity . . . 119
9.6 Conclusion . . . 120

Interlude 4: Ontology matching **121**

10 Alignment semantics **123**
10.1 Reminder: ontology semantics . . . 123
10.2 General alignment semantics . . . 124
10.3 Consistency, consequence and closure . . . 127
10.4 Local models from the standpoint of an ontology . . . 130
10.5 Conclusion . . . 131
10.6 Exercises . . . 132

11 Equalising semantics for alignments **133**
11.1 Conclusion 135

Interlude 5: Data interlinking **137**

12 Application: Distributed query evaluation **139**
12.1 Semantic peer-to-peer systems 139
12.2 The semantics of semantic peer-to-peer systems 140
12.3 Exercises 141
12.4 Conclusions 142

13 Conclusion **145**

IV Answers to exercises **147**

Part I

Graphs and ontologies

¡

Chapter 2

RDF: Resource description framework

The first issue raised on the semantic web is the expression of factual data on the web. To address it, a data expression language, adapted to the characteristics of the web, is required. The Resource Description Framework (RDF) is a W3C recommended language for expressing data on the semantic web [MANOLA and MILLER 2004]. It has been widely adopted in practice.

This chapter is devoted to the presentation of *Simple RDF*, i.e., RDF without specific (RDF or RDFS) vocabulary. We first recall (Section 2.2) its abstract syntax, its semantics (Section 2.3, using the notions of simple interpretations, models, simple entailment of, then Section 2.4 uses homomorphisms to characterize simple RDF entailment, instead of the equivalent interpolation lemma. Section 2.5 introduces the RDF entailment problem and its complexity.

2.1 IRI (and XML): the basic layer

IRIs (internationalized resource identifier) generalise URIs (uniform resource identifier [BERNERS-LEE et al. 1998]) which themselves generalize URLs (uniform resource locater) for identifying not only web pages but any resource (human, book, an author property). A IRI is meant to denote one precise resource, but a resource may have several IRIs.

The use of IRIs is specific to semantic web technologies, with respect to artificial intelligence practice. IRIs behave as simple identifiers. However, they are generally associated with a name space identifying their provenance (vocabulary or data source). For instance, `foaf:Person` identifies the `Person` concept in the friend-of-a-friend name space `http://xmlns.com/foaf/0.1/` shortened as `foaf:`. Thus an RDF graph may use different vocabularies with very little risks of conflict or ambiguity.

The semantic web relies loosely on XML. XML is only considered as a transport format. RDF is rather defined as an abstract object. A collection of RDF statements (RDF triples) can be intuitively understood as a directed labelled graph: resources are nodes and statements are arcs (from the subject node to the object node) connecting the nodes.

2.2 RDF Syntax

RDF can be expressed in a variety of formats. The initial formats were RDF/XML [CAROTHERS and SEABORNE 2014] and N-triple [GANDON and SCHREIBER 2014]. RDF 1.1 has also embraced various formats used for expressing RDF graphs: beyond RDF/XML and n-triples. The new well-defined formats are RDFa —enabling the embedding of RDF within HTML—, Turtle [BECKETT 2006], N-Quads —extending ntriples by quads—, and JSON-LD. These formats are usually supported by RDF data management systems, the so-called *triple stores*.

We use here its abstract syntax (triple format), which is sufficient for most purposes.

RDF terminology

To define the syntax of RDF, we need to introduce the *terminology* over which RDF graphs are constructed.

Definition 1 (RDF terminology [HAYES 2004])**.** *The RDF* terminology $\mathcal{T}$ *is the union of three pairwise disjoint infinite sets of* terms *:*

- *the set* $\mathcal{U}$ *of* IRIs*,*
- *the set* $\mathcal{L}$ *of* literals *(itself partitioned into two sets, the set* $\mathcal{L}_p$ *of* plain literals *and the set* $\mathcal{L}_t$ *of* typed literals*), and*
- *the set* $\mathcal{B}$ *of* variables*.*

The set $\mathcal{V} = \mathcal{U} \cup \mathcal{L}$ *of* names *is called the* vocabulary.

From now on, we use different notations for the elements of these sets: a variable will be prefixed by `?` (like `?b1`), a literal will be between quotation marks (like `"27"`), and the rest will be IRIs (like `foaf:Person` — `foaf:`[1] is a name space prefix used for representing personal information — `ex:friend` or simply `friend`).

To simplify notations, and without loss of generality, we do not distinguish here between simple and typed literals.

2.2.1 RDF graphs as sets of triples

RDF graphs are sets of triples ⟨subject, predicate, object⟩ constructed over IRIs, blanks, and literals [CARROLL and KLYNE 2004]. Because we want to stress the compatibility of the RDF structure with classical logic, we will use the term variable instead of that of "blank" which is a vocabulary specific to RDF. The specificity of blanks with regard to variables is their quantification. Indeed, a blank in RDF is a variable existentially quantified over a particular graph. We prefer to retain this classical interpretation which is useful when an RDF graph is placed in a different context.

Definition 2 (RDF graph)**.** *An* RDF triple *is an element of* $(\mathcal{U} \cup \mathcal{B}) \times \mathcal{U} \times \mathcal{T}$*. An* RDF graph *is a finite set of RDF triples.*

Excluding variables as predicates and literals as subject was an unnecessary restriction in the RDF design, that has been relaxed in RDF 1.1. These constraints complexify the syntax specification, and relaxing them neither changes RDF semantics nor the computational properties of reasoning. In consequence, we adopt such an extension introduced in [TER HORST 2005] and called *generalized RDF graphs*, or simply GRDF graphs.

Definition 3 (GRDF graph)**.** *A* GRDF triple *is an element of* $\mathcal{T} \times (\mathcal{U} \cup \mathcal{B}) \times \mathcal{T}$*. A GRDF graph is a finite set of GRDF triples.*

So, every RDF graph is a GRDF graph.

Example 1. *The following set of triples represents a GRDF graph:*

[1] `http://xmlns.com/foaf/spec/`

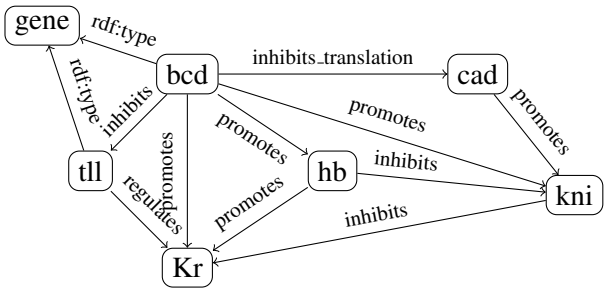


Figure 2.1: An RDF graph representing some interactions between genes within the early development of drosophila embryo.

$$
\begin{array}{l}
\{ \\
\quad \langle \texttt{?b1} \quad \texttt{foaf:name} \quad \texttt{"Faisal"} \rangle, \\
\quad \langle \texttt{?b1} \quad \texttt{ex:daughter} \quad \texttt{?b2} \rangle, \\
\quad \langle \texttt{?b2} \quad \texttt{?b4} \quad \texttt{?b3} \rangle, \\
\quad \langle \texttt{?b3} \quad \texttt{foaf:knows} \quad \texttt{?b1} \rangle, \\
\quad \langle \texttt{?b3} \quad \texttt{foaf:name} \quad \texttt{?name} \rangle \\
\}
\end{array}
$$

Intuitively, this graph means that there exists an entity named (`foaf:name`) `"Faisal"` that has a daughter (`ex:daughter`) that has some relation with another entity whose name is non determined, and that knows (`foaf:knows`) the entity named `"Faisal"`.

Notations In a triple $\langle s, p, o \rangle$, s is called the subject, p the predicate and o the object. If G is an RDF graph, we use $\mathcal{T}(G)$, $\mathcal{U}(G)$, $\mathcal{L}(G)$, $\mathcal{B}(G)$, $\mathcal{V}(G)$ to denote the set of terms, IRIs, literals, variables or names that appear in at least one triple of G. We denote by $subj(G)$ the set $\{s \mid \langle s, p, o \rangle \in G\}$ of elements appearing as a subject in a triple of a GRDF graph G. $pred(G)$ and $obj(G)$ are defined in the same way for predicates and objects. We call $nodes(G)$ the *nodes* of G, the set of elements appearing either as subject or object in a triple of G, i.e., $subj(G) \cup obj(G)$. A *term* of G is an element of $term(G) = subj(G) \cup pred(G) \cup obj(G)$. If $\mathcal{Y} \subseteq \mathcal{T}$ is a set of terms, we denote $\mathcal{Y} \cap term(G)$ by $\mathcal{Y}(G)$. For instance, $\mathcal{V}(G)$ is the set of names appearing in G.

A *ground* GRDF graph G is a GRDF graph with no variables, i.e., $term(G) \subseteq \mathcal{V}$ (see Example 2).

Example 2 (Ground (G)RDF graph)**.** *RDF can be used for exposing a large variety of data. For instance, Figure 2.1 shows the RDF graph representing part of the gene regulation network acting in the drosophila embryo. Nodes represent genes and properties express regulation links, i.e., the fact that the expression of the source gene has an influence on the expression of the target gene. The triples of this graph are the following:*

```
dm:bcd rdf:type rn:gene.
dm:bcd rn:inhibits_translation dm:cad.
dm:bcd rn:promotes dm:hb.
dm:bcd rn:promotes dm:kni.
dm:bcd rn:promotes dm:Kr.
dm:bcd rn:inhibits dm:tll.
dm:cad rn:promotes dm:kni.
dm:hb rn:inhibits dm:kni.
dm:hb rn:promotes dm:Kr.
dm:kni rn:inhibits dm:Kr.
dm:tll rn:regulates dm:Kr.
dm:tll rdf:type rn:gene.
```

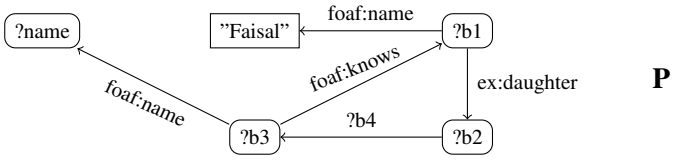


Figure 2.2: A GRDF graph.

This example uses only IRIs.

2.2.2 Graph representation of RDF triples

A *simple GRDF graph* can be represented graphically as a directed labeled multigraph $\langle N, E, \gamma, \lambda \rangle$ where the set of nodes N is the set of terms appearing as a subject or object in at least one triple of G, the set of arcs E is the set of triples of G, γ associates to each arc a pair of nodes (its extremities) $\gamma(e) = \langle \gamma_1(e), \gamma_2(e) \rangle$ where $\gamma_1(e)$ is the source of the arc e and $\gamma_2(e)$ its target; finally, λ labels the nodes and the arcs of the graph: if s is a node of N, i.e., a term, then $\lambda(s) = s$, and if e is an arc of E, i.e., a triple (s, p, o), then $\lambda(e) = p$, i.e., if $\langle s, p, o \rangle$ is a triple, then $s \xrightarrow{p} o$ is an arc. When drawing such graphs, the nodes resulting from literals are represented by rectangles while the others are represented by rectangles with rounded corners.

For example, the GRDF triples given in Example 1 can be represented graphically as in Figure 2.2.

RDF 1.1 has acknowledged the notion of *RDF Dataset* which is a collection of RDF graphs and that of *named graphs* which are graphs identified by IRIs. This identification allows to assert statements (triples) about graphs and hence triples. Such statements may be about the origin of the graph or the confidence in what it asserts. This function was previously fulfilled by '*quads*', i.e., triples which were usually tagged by a IRI.

In what follows, we do not distinguish between the two views of the RDF syntax (as sets of triples or directed labeled graphs). We will then speak interchangeably about their nodes, their arcs, or the triples which make them up.

2.2.3 Logical representation of RDF triples

Some RDF graphs may be translated as formulas in a positive (without negation), conjunctive, existential and function-free first-order logic. These are those graphs in which predicates are not used as subject or object. To each triple $\langle s, p, o \rangle$ corresponds the atomic formula $p(s, o)$, such that p is a predicate name, and o and s are constants if these elements are IRIs or literals, and variables otherwise. A graph is translated as the existential closure of the conjunction of atomic formulas associated to its triples. Hence, the graph of Figure 2.2 with ?b4 replaced by rel:worksWith is translated by:

$$\exists ?b1, ?b2, ?b3, ?name; \texttt{foaf:name}(?b1, "Faisal") \wedge \texttt{rel:daughter}(?b1, ?b2) \wedge \texttt{rel:worksWith}(?b2, ?b3) \wedge \texttt{foaf:name}(?b3, ?name) \wedge \texttt{foaf:knows}(?b3, ?b1)$$

The models of such a formula are isomorphic to those of the simple RDF semantics of the graph (we state it without proof).

2.3 Simple RDF Semantics

[HAYES 2004] introduces several semantics for RDF graphs. In this section, we present only the *simple semantics* without RDF/RDFS vocabulary [BRICKLEY and GUHA 2004]. The definitions of interpretations, models, satisfiability, and entailment correspond to the *simple interpretations*, *simple models*, *simple satisfiability*, and *simple entailments* of [HAYES 2004]. The semantics of an RDF graph is defined in model theory [HAYES and PATEL-SCHNEIDER 2014]. RDF is thus a proper logics. RDF and RDFS consequences (or entailments) can be polynomially reduced to simple entailment via RDF or RDFS rules [BAGET 2005; TER HORST 2005] (see Section 3.1.3).

The presentation here is mostly based on [HAYES 2004] but could be reconsidered with respect to [HAYES and PATEL-SCHNEIDER 2014].

2.3.1 Interpretations

An interpretation describes possible way(s) the world might be in order to determine the truth-value of any ground RDF graph. It does this by specifying the denotation of each IRI (and each literal). In addition, if a IRI is used to indicate a property, it specifies which things in the universe are related by this property.

Interpretations that assign particular meanings to some names in a given vocabulary will be named from that vocabulary, e.g., RDFS interpretations (see Section 3.1). An interpretation with no particular extra conditions on a vocabulary (including the RDF vocabulary itself) will be simply called an interpretation.

Definition 4 (Interpretation of a vocabulary)**.** *Let $V \subseteq \mathcal{V} = \mathcal{U} \cup \mathcal{L}$ be a vocabulary, an* interpretation *of V is a quadruple $I = \langle I_R, I_P, I_{EXT}, \iota \rangle$ such that:*

- *I_R is a set of* resources *that contains $V \cap \mathcal{L}$;*
- *$I_P \subseteq I_R$ is a set of properties;*
- *$I_{EXT} : I_P \to 2^{I_R \times I_R}$ associates to each property a set of pairs of resources called the* extension *of the property;*
- *the interpretation function $\iota : V \to I_R$ associates to each name in V a resource of I_R, if $v \in \mathcal{L}$, then $\iota(v) = v$.*

This semantics has this peculiarity that predicates, which naturally corresponds to classical dyadic predicates, may also be considered as resources. Hence, the triple $\langle$`rdf:type`, `rdf:type`, `rdf:Property`$\rangle$ is legitimate in RDF and can be interpreted (it indeed means that `rdf:type` denotes a predicate). This is achieved by interpreting triples in two steps: a first step (ι) associates a denotation to each IRI used and a second step (I_{EXT}) interprets those used in predicate position as binary relations. This is the main peculiarity of the RDF semantics with respect to that of first-order logics.

2.3.2 Models

By providing RDF with formal semantics, [HAYES 2004; HAYES and PATEL-SCHNEIDER 2014] expresses the conditions under which an interpretation is a model for an RDF graph. The usual notions of validity, satisfiability and consequence are entirely determined by these conditions.

Intuitively, a ground triple $\langle s, p, o \rangle$ in a GRDF graph will be true under the interpretation I if p is interpreted as a property (for example, r_p), s and o are interpreted as resources (for example, r_s and r_o, respectively), and the pair of resources $\langle r_s, r_o \rangle$ belongs to the extension of the property r_p. A triple $\langle s, p, ?b \rangle$ with the variable $?b \in \mathcal{B}$ would be true under I if there exists a resource r_b such that the pair $\langle r_s, r_b \rangle$ belongs to the extension r_p. When interpreting a variable node, an arbitrary resource can be chosen. To ensure that a variable always is interpreted by the same resource, extensions of the interpretation function is defined as follow.

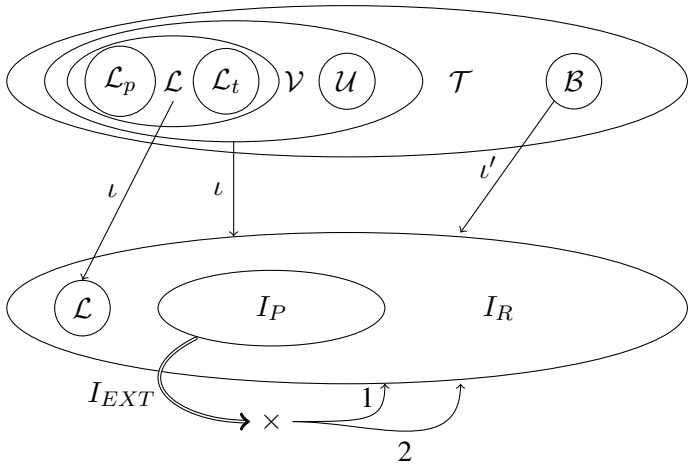


Figure 2.3: Domain structure for Simple RDF semantics.

Definition 5 (Extension to variables)**.** *Let $I = \langle I_R, I_P, I_{EXT}, \iota\rangle$ be an interpretation of a vocabulary $V \subseteq \mathcal{V}$, and $B \subseteq \mathcal{B}$ a set of variables. An extension of ι to B is a mapping $\iota' : \mathcal{V} \cup B \to I_R$ such that $\forall x \in V$, $\iota'(x) = \iota(x)$.*

An interpretation I is a model of GRDF graph G if all triples are true under I.

Definition 6 (Model of a GRDF graph)**.** *Let $V \subseteq \mathcal{V}$ be a vocabulary, and G be a GRDF graph such that every name appearing in G is also in V ($\mathcal{V}(G) \subseteq V$). An interpretation $I = \langle I_R, I_P, I_{EXT}, \iota\rangle$ of V is a* model *of G iff there exists an extension ι' that extends ι to $\mathcal{B}(G)$ such that for each triple $\langle s, p, o\rangle$ of G, $\iota'(p) \in I_P$ and $\langle \iota'(s), \iota'(o)\rangle \in I_{EXT}(\iota'(p))$. The mapping ι' is called a* proof *of G in I.*

2.3.3 Satisfiability, validity, entailment and consequence

The following definition is the standard model-theoretic definition of satisfiability validity and consequence.

Definition 7 (Satisfiability, validity, consequence)**.** *A graph G is* satisfiable *iff it has a model. G is* valid *iff for every interpretation I of a vocabulary $V \supseteq \mathcal{V}(G)$, I is a model of G. A graph G' is a* consequence *of a graph G, or G* entails *G', denoted $G \models_{GRDF} G'$, iff every model of G is also a model of G'.*

Proposition 1 (Empty graph, subgraph, instance lemmata [HAYES 2004])**.**

Empty graph lemma *the empty set of triple is entailled by any graph and does not entail any graph but itself;*

Subgraph lemma *a graph entails all its subgraphs (i.e., subsets of triples);*

Instance lemma *a graph is entailled by any of its instances (i.e., variables substitued by values);*

Sketch of proof. The subgraph lemma holds because any model of a subgraph imposes less constraints on interpretations that the graph it is a subgraph of. Hence, any of its interpretation will be a interpretation of the supergraph. This also entails the Empty lemma, because the empty graph is a subgraph of any graph. Concerning the Instance lemma, the instantiated graph imposes more constraints than the initial graph, because the assignment of the corresponding variable is fixed. Hence, any model of the instance is a

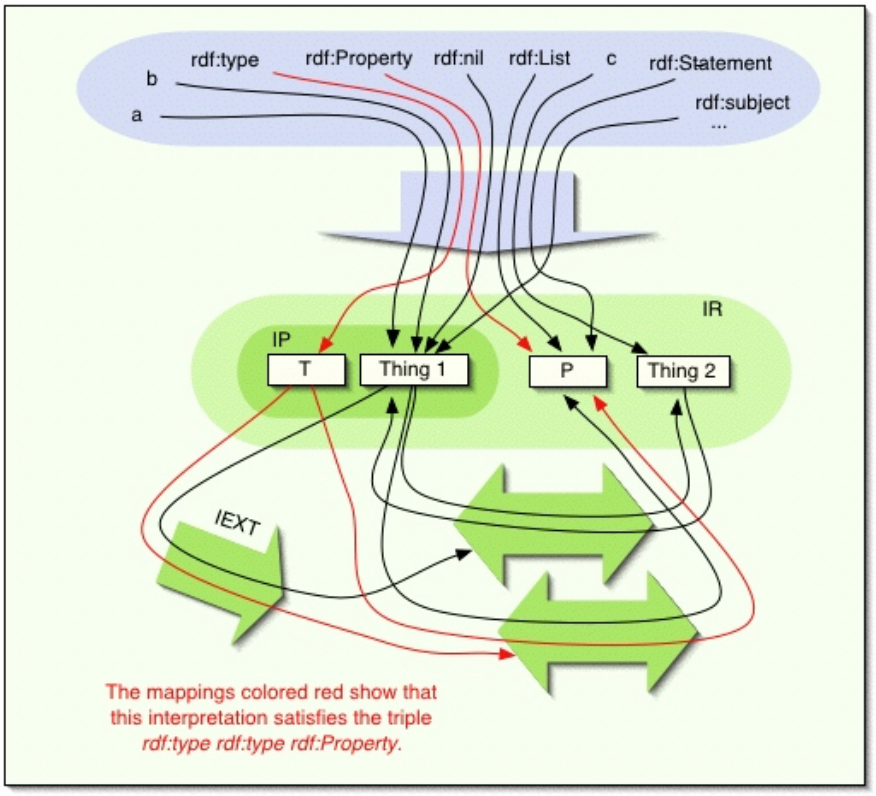


Figure 2.4: Example of interpretation of the domain structure (from [HAYES 2004]).

model of the general graph (with the extension to blank assigning the instantiated value to each instantiated variable). Finally, any graph is a subgraph of its union with other graphs which, by the Subgraph lemma, makes it entailed. □

Currently the two following propositions rely on the same construct: the Herbrand model or isomorphic model.

Proposition 2 (Satisfiability, validity [BAGET 2005; TER HORST 2005])**.** *Every GRDF graph is satisfiable. The only valid GRDF graph is the empty graph.*

Proof. (Satisfiability) See proof of Proposition 3.

(Validity) a non empty GRDF graph has no proof in an interpretation in which all properties are interpreted by I_{EXT} as an empty set. □

Proposition 3 (Interpolation lemma [HAYES 2004])**.**

$$G \models_{GRDF} H \textit{ iff a subgraph of } G \textit{ is an instance of } H$$

Proof. $\Leftarrow$: This is clearly true due to the subgraph and instance lemmas: if G' is a subgragph of G, then $G \models G'$ because all models of G trivially satisfy the triples of G' (subgraph lemma). If G' is an instance of H, then $G' \models H$ because, for each model $I = \langle I_R, I_P, I_{EXT}, \iota \rangle$ of G', there exist an extension to variables ι' satisfying all triples and for each triple $\langle s', p', o' \rangle \in G'$ instantiating a triple $\langle s, p, o \rangle \in H$, there is an extension of ι'' to variables, such that for each element x in the triple, $\iota''(x) = \iota'(x')$ if $x \in \mathcal{B}(H) \setminus \mathcal{G}'$, and $\iota''(x) = \iota'(x)$ otherwise (instance lemma).

$\Rightarrow$: The Herbrand interpretation $\hat{I}^G$ of the graph G is defined by $\hat{I}^G = \langle \hat{I}^G_R, \hat{I}^G_P, \hat{I}^G_{EXT}, \hat{\iota}^G \rangle$ such that:

- $\hat{I}^G_R = term(G)$;
- $\hat{I}^G_P = pred(G)$;
- $\hat{I}^G_{EXT} : \hat{I}^G_P \to 2^{\hat{I}^G_R \times \hat{I}^G_R}$ such that $\forall p \in \hat{I}^G_P$, $\hat{I}^G_{EXT}(p) = \{\langle s, o \rangle \in \hat{I}^G_R \times \hat{I}^G_R ; \langle s, p, o \rangle \in G\}$;
- the interpretation function $\hat{\iota}^G : V \to \hat{I}^G_R$ is the identity function over $term(G)$.

$\hat{I}^G$ is clearly a model of G, since it can be extended by the identity function on blanks and trivially satisfies all constraints ($\forall x \in term(G)$, $\iota'(x) = \iota(x) = x$). The condition of Definition 6 immediately follows from this construction. This proves the satisfiability part of Proposition 2.

If $G \models_{GRDF} H$, then $\hat{I}^G$ is also a model of H. This means that there exists an extension $\hat{\iota}^H$ of $\hat{\iota}^G$ to blanks satisfying every triple in H. Hence, there must exists a map $\hat{\iota}^H : \mathcal{B}(H) \cup \mathcal{V}(G) \to \hat{I}^G_R$, such that for any triple $\langle s, p, o \rangle \in H$, $\langle \hat{\iota}^H(s), \hat{\iota}^H(o) \rangle \in \hat{I}^G_{EXT}(\hat{\iota}^H(p))$. But given the definition of $\hat{I}^G_{EXT}$, this means that $\langle \hat{\iota}^H(s), \hat{\iota}^H(p), \hat{\iota}^H(o) \rangle \in G$. For each element x of this triple, either $x \in \mathcal{V}(G)$ (and then $\hat{\iota}^H(x) = x$) or $x \in \mathcal{B}(H)$ (and then $\hat{\iota}^H(x) \in term(G)$). This triple is thus necessarily an instance of $\langle s, p, o \rangle$. So, $\forall \langle s, p, o \rangle \in H$, there exists an instantiation $\langle s', p', o' \rangle \in G$, of it, which proves that a subgraph of G is an instantiation of H. □

2.4 Inference mechanism

SIMPLE RDF ENTAILMENT [HAYES 2004] can be characterized as a kind of graph homomorphism. A *graph homomorphism* from an RDF graph H into an RDF graph G, as defined in [BAGET 2005; GUTIERREZ et al. 2004], is a mapping σ from the nodes of H into the nodes of G preserving the graph structure, i.e., for each node $x \in H$, if $\lambda(x) \in \mathcal{U} \cup \mathcal{L}$ then $\lambda(\sigma(x)) = \lambda(x)$; and each edge $x \xrightarrow{p} y$ is mapped to $\sigma(x) \xrightarrow{\sigma(p)} \sigma(y)$. This definition is similar to the projection used to characterize entailment of conceptual graphs [CHEIN and MUGNIER 2009] (see [CORBY et al. 2000] for a precise relationship between RDF and

conceptual graphs). We modify this definition to the one that maps $term(H)$ into $term(G)$. Maps are used to ensure that a variable always mapped to the same term, as done for extensions to interpretations.

Definition 8 (Map)**.** *Let $V_1 \subseteq \mathcal{T}$, and $V_2 \subseteq \mathcal{T}$ be two sets of terms, a* map *from V_1 to V_2 is a mapping $\sigma : V_1 \to V_2$ such that $\forall x \in (V_1 \cap \mathcal{V})$, $\sigma(x) = x$.*

Obviously, such a map exists only if $V_1 \cap \mathcal{V} \subseteq V_2 \cap \mathcal{V}$. An RDF homomorphism is a map preserving the graph structure.

Definition 9 (GRDF homomorphism)**.** *Let G and H be two GRDF graphs. A GRDF homomorphism from H into G is a map σ from $term(H)$ to $term(G)$ such that $\forall \langle s, p, o\rangle \in H$, $\langle \sigma(s), \sigma(p), \sigma(o)\rangle \in G$.*

[GUTIERREZ et al. 2004] provides without proof an equivalence theorem (Theorem 3) between RDF entailment and maps. A proof is provided in [BAGET 2005] also for RDF graphs, but the homomorphism involved is a mapping from nodes to nodes, and not from terms to terms. In RDF, the two definitions are equivalent. However, the terms-to-terms version is necessary to extend the theorem of RDF (Theorem 4) to the PRDF graphs studied in [ALKHATEEB, BAGET, et al. 2009]. All the proofs are available in [ALKHATEEB 2008].

Example 3 (GRDF homomorphism)**.** *Figure 2.5 shows two GRDF graphs Q and G (note that the graph Q is the graph P of Figure 2.2, to which the following triple is added $\langle$ `?b3, foaf:mbox, ?mbox`$\rangle$. The map σ_1 defined by $\{$(`"Faisal", "Faisal"`), (`?b1, ?c1`), (`?name, "Natasha"`), (`?b2, ?c2`) , (`?b4, ex:friend`), (`?mbox, "natasha@yahoo.com"`), (`?b3, ex:Person1`)$\}$ is a GRDF homomorphism from Q into G. And the map σ_2 defined by $\{$(`"Faisal", "Faisal"`), (`?b1, ?c1`), (`?name, "Deema"`), (`?b3, ex:Person2`), (`?b4, ex:friend`), (`?b2, ?c2`)$\}$ is a GRDF homomorphism from P into G. Note that σ_2 cannot be extended to a GRDF homomorphism from Q into G since there is no mailbox for `"Deema"` in G.*

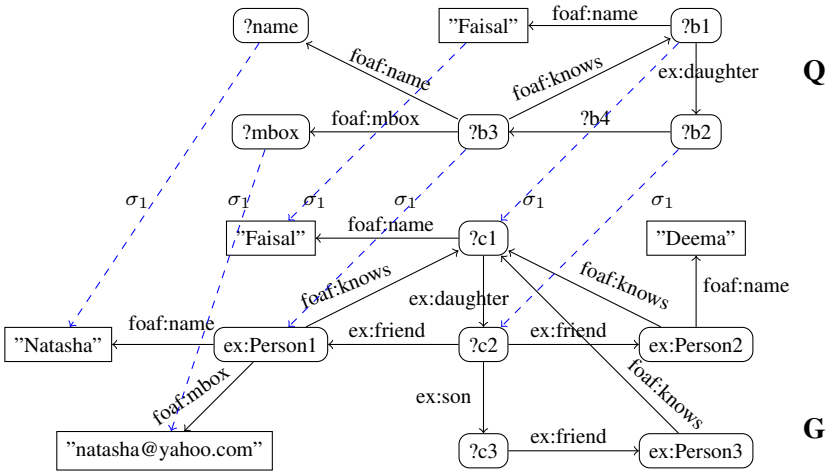


Figure 2.5: A GRDF homomorphism from Q into G.

Theorem 4. *Let G and H be two GRDF graphs, then $G \models_{GRDF} H$ if and only if there is a GRDF homomorphism from H into G.*

Proof. The proof of this theorem is left as an exercise (it is given in [ALKHATEEB 2008]). It relies on showing that an homomorphism from H into G determines a subgraph of G (the image of H in G by the homomorphism) and that this subgraph is necessarily an instance of H by Definition 9. Then, by Proposition 3, the equivalence is shown. □

This equivalence between the semantic notion of entailment and the syntactic notion of homomorphism is the ground by which a correct and complete query answering procedure can be designed. More precisely, the set of answers to a GRDF graph query Q over an RDF knowledge base G are the set of RDF homomorphisms from Q into G which, by Theorem 4, correspond to RDF consequence.

2.5 Computational properties

The decision problem associated to simple RDF semantics is called SIMPLE RDF ENTAILMENT, and is defined as follows:

SIMPLE (G)RDF ENTAILMENT
Instance: two GRDF graphs G and H.
Question: Does $G \models_{\text{GRDF}} H$?

SIMPLE (G)RDF ENTAILMENT is an NP-complete problem for RDF graphs [GUTIERREZ et al. 2004]. For GRDF graphs, its complexity remains unchanged [PÉREZ et al. 2009].

Hint of proof. Graph matching, an NP-complete problem, may be reduced to entailment within RDF graphs with all nodes of the source graph with a different IRI and all nodes of the target graph blanks and the same predicate on each edge. □

Polynomial subclasses of the problem have been exhibited based upon the structure or labeling of the query H:

- when the query H is ground [TER HORST 2005], or more generally if it has a bounded number of variables,
- when the query H is a tree or admits a bounded decompositions into a tree, according to the methods in [GOTTLOB et al. 1999] as shown in [BAGET 2005].

2.6 Conclusion

RDF (or Simple RDF) is a language allowing for expressing information as a graph in which resources are related by predicates. This language has a model theoretic semantics allowing for defining the notion of consequence (or entailment) between such graphs. Moreover, entailment is equivalent to the existence of an homomorphism between the entailed graph and the entailing graph. In its generality, checking for entailment is an NP-complete problem.

This allows to check if a formula (query) is a consequence of the information expressed in an RDF graph.

2.7 Exercises

Exercise 1 (RDF assertions)**.** *Consider the four graphs of Figure 2.6.*

1. *Are they all well-formed RDF graphs? Why?*
2. *Express the graph of Figure 2.6(a) as a set of triples. You will list the sets of literals, IRIs and variables (or blanks) found in this graph.*
3. *Give an informal meaning of this graph or its expression in the predicate calculus*

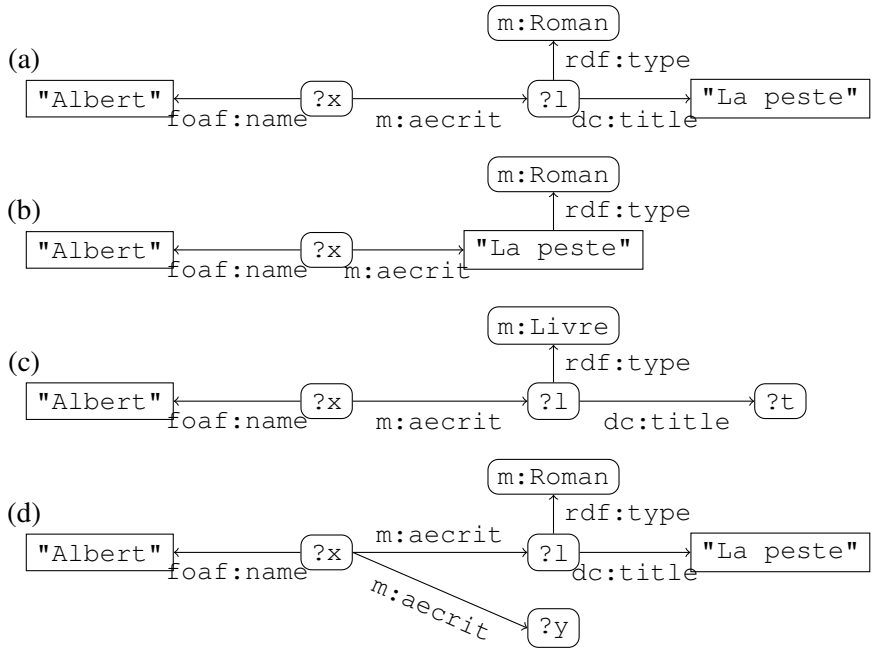


Figure 2.6: RDF graphs.

4. *Consider the RDF graphs of Figure 2.6 which are well-formed, do some of them entail others? Explain why.*

Exercise 2 (RDF and assertions)**.** *Consider the two graphs of Figure 2.7 dealing with the expression of social relationships.*

1. *Express the graph of Figure 2.7(b) as a set of triples. You will give the list of litterals, IRIs and variables (or blanks) in this graph.*
2. *What is, informally, the meaning of the graph of Figure 2.7(b) (tell it in English or French or predicate calculus)?*
3. *Does one of these graphs entail the other? (explain why)*

Exercise 3 (RDF graphs)**.** *Here are the 8 triples of an RDF graph G about writers and their works: (all identifiers correspond in fact to IRIs, _:b is a blank node):*

$$\langle \texttt{d:Poe}, \texttt{o:wrote}, \texttt{d:TheGoldBug}\rangle \ \langle \texttt{d:Baudelaire}, \texttt{o:translated}, \texttt{d:TheGoldBug}\rangle$$
$$\langle \texttt{d:Poe}, \texttt{o:wrote}, \texttt{d:TheRaven}\rangle \ \langle \texttt{d:Mallarmé}, \texttt{o:translated}, \texttt{d:TheRaven}\rangle$$
$$\langle \texttt{d:TheRaven}, \texttt{rdf:type}, \texttt{o:Poem}\rangle \ \langle \texttt{d:Mallarmé}, \texttt{o:wrote}, \texttt{\_:b}\rangle$$
$$\langle \texttt{\_:b}, \texttt{rdf:type}, \texttt{o:Poem}\rangle \ \langle \texttt{d:TheGoldBug}, \texttt{rdf:type}, \texttt{o:Novel}\rangle$$

1. *Draw an RDF graph corresponding to these statements*
2. *Express in English the meaning of these statements.*

Exercise 4 (RDF Manipulation)**.** *Consider the following statements composing the RDF graph G:*

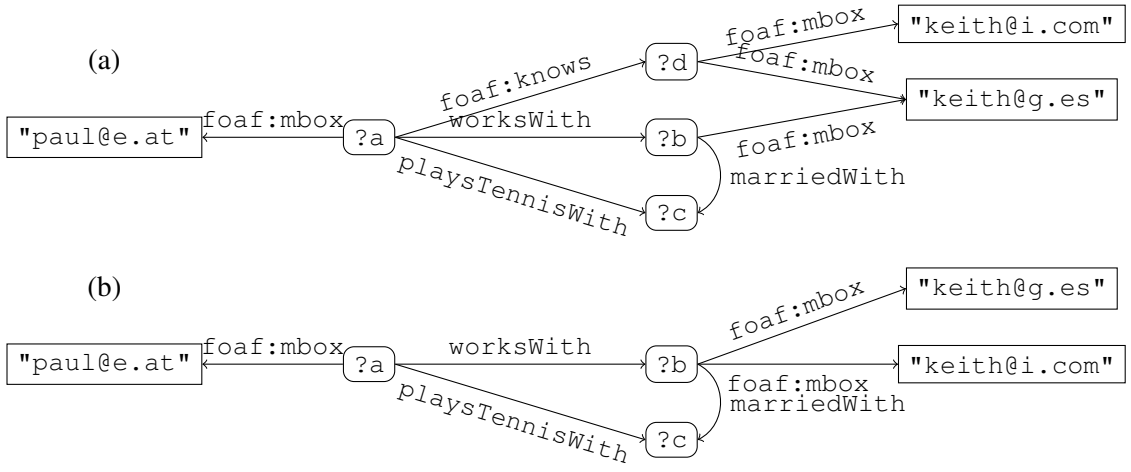


Figure 2.7: RDF graphs.

```
ex:book1 rdf:type mr:Book .
ex:book1 dc:title "For whom the bell tolls" .
ex:book1 dc:date 1940 .
ex:book1 dc:creator ex:eh .
ex:book1 mr:storedIn "Living room" .

ex:book2 rdf:type mr:Book .
ex:book2 dc:title "Pour qui sonne le glas" .
ex:book2 mr:translationOf ex:book1 .
ex:book2 dc:creator ex:eh .
ex:book2 dc:date 1948 .
ex:book2 dc:publisher ex:gal .

ex:movie1 rdf:type mr:Movie .
ex:movie1 dc:title "For whom the bell tolls" .
ex:movie1 mr:adaptedFrom ex:book1 .
ex:movie1 mr:director ex:sw .
ex:movie1 mr:cast ex:ib .
ex:movie1 mr:cast ex:gc .
ex:movie1 mr:storedIn "Computer drive" .
ex:movie1 dc:date 1943 .

ex:eh rdf:type foaf:Person .
ex:eh foaf:name "Ernest Hemingway" .

ex:sw rdf:type foaf:Person .
ex:sw foaf:name "Sam Wood .

ex:ib rdf:type foaf:Person .
ex:ib foaf:name "Ingrid Bergman" .

ex:gc rdf:type foaf:Person .
ex:gc foaf:name "Gary Cooper" .
```

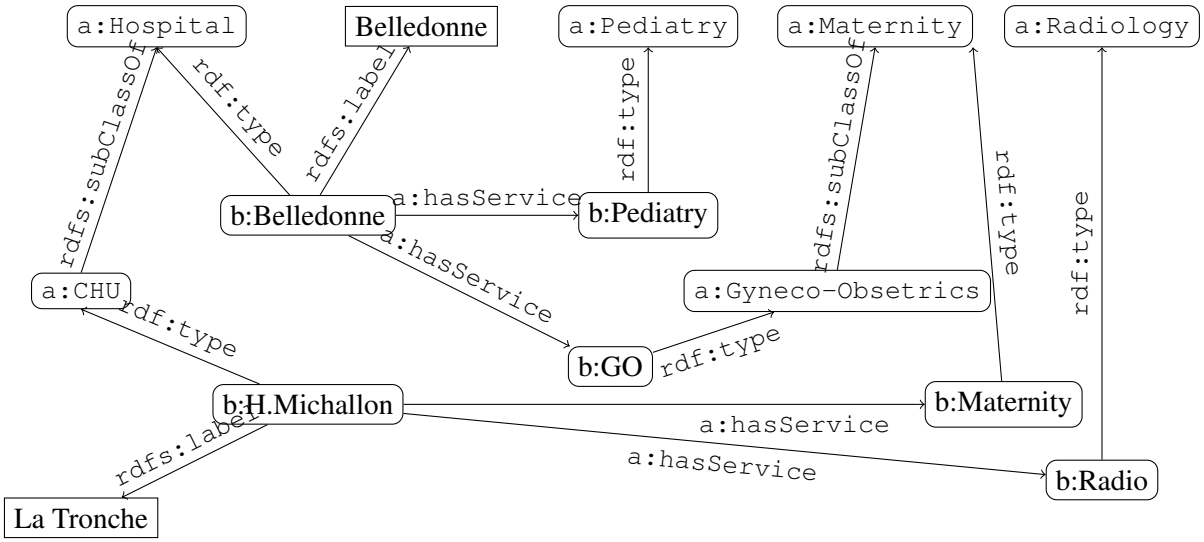


Figure 2.8: RDF graph G.

```
ex:gal rdf:type foaf:Organization .
ex:gal foaf:name "Gallimard" .
```

1. *Which different classes are there in graph G? Why are they classes?*
2. *Express G as a graph in graphical form.*

Consider the following as the RDF graph G' (`_:` are blank identifiers):

```
_:x rdf:type mr:Movie .
_:x mr:adaptedFrom _:y .
_:y dc:creator _:p .
_:p foaf:name "Ernest Hemingway" .
```

3. *Paraphrase the graph G'.*
4. *Is G' entailed by G? Detail why?*

Exercise 5 (RDF Entailment)**.** *Consider the graph G of Figure 2.8.*

1. *Explain what it means (in English or French, taking into account the meaning of RDFS vocabulary).*
2. *Rewrite it as a set of triples.*

Given the GRDF graph

P={ *?x* `rdf:type a:CHU`. *?x* `a:hasService` *?y*. *?y* `rdf:type a:Maternity`. }

3. *Does $G \models_{RDF} P$? (explain how and/or why)*

Exercise 6 (RDF Entailment)**.** *We have two graphs coming from two different sources. Consider the graph G_1 coming from the municipality made of the following triples:*

```
a:Pierre o1:father a:Carole .          a:Marie o1:mother a:Carole .
a:Pierre o1:father a:Kevin .           a:Marie o1:mother a:Kevin .
a:Jacques o1:father a:Jean .           a:Marie o1:mother a:Jean .
_:b1 o1:father a:Sylvie .              a:Stephanie o1:mother a:Sylvie .
_:b1 o1:father a:William .             a:Stephanie o1:mother a:William .
a:Jacques o1:father a:Julie .          a:Nabila o1:mother a:Julie.
a:Sven o1:father a:Laurent .           a:Lucie o1:mother a:Laurent .
```

and G_2 coming from the school made of the following information:

```
a:Carole o2:attendsClass b:4e3.        a:Carole rdf:type o2:Female .
a:Kevin o2:attendsClass b:6e1 .        a:Kevin rdf:type o2:Male .
a:Sylvie o2:attendsClass b:5e2 .       a:Sylvie rdf:type o2:Female .
a:William o2:attendsClass b:5e2 .      a:William rdf:type o2:Male .
a:Julie o2:attendsClass b:5e2 .        a:Julie rdf:type o2:Female .
a:Laurent o2:attendsClass b:4e3  .     a:Laurent rdf:type o2:Male .
a:Jasmine o2:attendsClass b:5e1 .      a:Jasmine rdf:type o2:Female .
```

1. *Draw these two graphs (together);*
2. *In order, to work with these two graphs, we want to answer queries that span through both of them. Consider the following graph Q_1:*

   ```
   _:x o2:attendsClass _:w .
   _:y o2:attendsClass _:w .
   _:x rdf:type o2:Male .
   _:y rdf:type o2:Female .
   _:z o:parent _:x .
   _:z o:parent _:y .
   ```

   Express in English the meaning of Q_1. Is Q_1 entailed by any of G_1 or G_2? (explain why)
3. *Express the graph Q_2 corresponding to the English: "there exist two people sharing at least one parent attending the same class"? Does $Q_2 \models_{RDF} Q_1$ or $Q_1 \models_{RDF} Q_2$?*

Exercise 7 (RDF Interpretation and entailment)**.** *Consider the graph G describing holiday packages:*

```
_:b1 rdf:type o:Package .              d:PousadaDesArts rdf:type o:Pousada .
_:b1 o:destination d:Salvador .        _:b1 o:activity _:b2 .
_:b1 o:accomodation d:PousadaDesArts . _:b2 rdf:type o:Swimming .

_:b3 rdf:type o:Package .              d:Metropol rdf:type o:GrandHotel .
_:b3 o:destination d:Moskow .          _:b3 o:activity d:VolgaCruise .
_:b3 o:accomodation d:Metropol .       d:VolgaCruise rdf:type o:Cruise .

_:b4 rdf:type o:Package .              d:ToyofukuRyokan rdf:type o:Ryokan .
_:b4 o:destination d:Kobe .            _:b4 o:activity _:b5 .
_:b4 o:accomodation d:ToyofukuRyokan . _:b5 rdf:type o:SwordFighting .
```

1. *Draw the graph G.*
2. *Define an RDF-interpretation $\mathcal{I}$ of G's vocabulary ($V(G)$).*
3. *Given the following graph H:*

```
_:x rdf:type o:Package .
_:x o:accomodation _:acc .
_:x o:activity _:act .
```

Does your interpretation satisfies H (said otherwise, is $\mathcal{I}$ a model of H)?

4. *Does $G \models H$? Show it.*
5. *Given the following graph K:*

```
_:y rdf:type o:Package .
_:y o:accomodation _:acc .
_:acc rdf:type o:Local .
_:y o:activity _:act .
_:act rdf:type o:Sport .
```

Does $G \models K$? Tell why.

The web of data

In 2006, from the observation that the semantic web development was lagging due to the lack of resources, Tim Berners-Lee put forth a methodology for publishing large quantities of data on the web so that they can help the semantic web [BERNERS-LEE 2006].

This defined the '*web of data*' through four principles for publishing data on the web, summarised as [BERNERS-LEE 2006; HEATH and BIZER 2011]:

1. Resources are identified by IRIs.
2. IRIs are dereferenceable, i.e., can be looked up on the web.
3. When a IRI is dereferenced, a description of the identified resource should be returned, ideally adapted through content negotiation.
4. Published web data sets must contain links to other web data sets.

Although not explicitly specified, linked data sources are more usable if they are published with semantic web technologies: IRIs for identifying resources, RDF for describing them, OWL for defining the used vocabularies, SPARQL for accessing data, and asserting links between sources through the `owl:sameAs` predicate.

The 'five stars rating', to be interpreted as incremental steps, has been introduced for measuring how much of this usability is achieved, namely:

- ⋆ Publish data on the web in any format, e.g., a scan of a table in PDF;
- ⋆⋆ Use structured data formats, e.g., a table in Excel instead of its scan in PDF;
- ⋆⋆⋆ Use non-proprietary formats, e.g., CSV instead of Excel, such that users have direct access to the raw data;
- ⋆⋆⋆⋆ Use universal formats to represent data, such as RDF, which encapsulates both syntax and semantics;
- ⋆⋆⋆⋆⋆ Link data to other data sets on the web, thereby providing context.

The first three stars are easy to reach and these already enable some data reuse. However, humans still have to handle all the semantic issues related to integration. In order to have data that is more easily discoverable and interoperable, it is necessary to reach the fourth and the fifth stars.

The web of data is thus a huge RDF graph reachable through the HTTP protocol like the web. Very quickly it developed around DBpedia [BIZER et al. 2009], a massive extraction of Wikipedia in RDF. Many efforts linked their resources to DBpedia. Soon after, several governments encouraged the publication of their (open) data in RDF. Since then, libraries, museums, research institutions of various kinds joined the linked open data cloud. In 2016, it contained 9960 data sources providing more than 150 billion triples using 576 vocabularies or ontologies[2].

[2]Sources: `http://stats.lod2.eu/` and `http://lov.okfn.org`.

Chapter 3

Ontology languages

Once able to express data on the web, the next problem is the definition of the vocabulary used in the RDF graphs. This means listing the terms used to describe data as well as expressing the axioms constraining the use of such terms allowing a machine to interpret them properly. Such vocabularies are defined as *ontologies*. We will consider below ontologies as sets of axioms in a specific logic. We discuss the main ontology languages designed for the web: RDF Schema (§3.1) and OWL (§3.2).

3.1 RDF Schema

What we considered so far is only the Simple RDF semantics. Full RDF is defined by identifying a particular vocabulary, the RDF vocabulary, and adding constraints to the definition of model in relation with this vocabulary. Because these changes are minor, we do not consider them and instead directly define RDFS Schema which contains full RDF.

RDFS (RDF Schema) [BRICKLEY and GUHA 2004] is an extension of RDF designed to describe relationships between resources using a set of reserved IRIs called the RDFS vocabulary. In the above example, the reserved word `rdf:type` can be used to relate instances to classes, e.g., `book1` is of type `publication`.

This section focusses on RDF and RDFS as extensions of the Simple RDF language presented in Chapter 2. Both extensions are defined in the same way:

- They consider a particular set of IRIs of the vocabulary prefixed by `rdf:` and `rdfs:`, respectively.
- They add additional constraints to the resources associated to these terms in interpretations.

In adding new constraints to RDFS interpretations, RDFS documents may have less models, and thus more consequences. It is possible for example, in RDF, to deduce $\langle$`ex:author rdf:type rdf:Property`$\rangle$ from $\langle$`ex:person1 ex:author "Alkhateeb"`$\rangle$; in RDFS, to deduce $\langle$`ex:document1 rdf:type ex:Biography`$\rangle$ from $\{\langle$`ex:document1 rdf:type rdf:Autobiography`$\rangle$, $\langle$`ex:Autobiography rdfs:subClassOf ex:Biography`$\rangle\}$.

As usual, we first present the vocabulary (§3.1.1) and the constraints they put on the interpretation of the language (§3.1.2). We also introduce a "normative" inference mechanism (§3.1.3).

3.1.1 RDFS as an RDF vocabulary

In RDF and RDF Schema, there exists a set of reserved words, the RDF and RDFS vocabularies designed to describe relationships between resources like classes, e.g., `geo:City subClassOf geo:PopulatedArea`, and relationships between properties, e.g., `country subPropertyOf is-contained-in`.

RDF already introduced a few keywords in the `rdf:` name space for structuring knowledge:

RDF Typing vocabulary		
rdf:type[type]	rdf:Property[prop]	rdf:XMLLiteral[xmlLit]
Reification vocabulary		
rdf:Statement[stat]		
rdf:subject[subj]	rdf:predicate[pred]	rdf:object[obj]
Container vocabulary		
rdf:first[first]	rdf:rest[rest]	rdf:nil[nil]
rdf: 1[1]	rdf: 2[2]	rdf: i[i]
rdf:List[list]	rdf:Bag[bag]	rdf:Seq[seq]
rdf:Alt[alt]		
Miscellaneous vocabulary (do not use)		
rdf:value[value]		
RDFS Typing vocabulary		
rdfs:Class[class]	rdfs:Resource[res]	rdfs:Literal[literal]
rdfs:domain[dom]	rdfs:range[range]	rdfs:Container[cont]
rdfs:subClassOf[sc]	rdfs:subPropertyOf[sp]	rdfs:Datatype[datatype]
Container vocabulary		
rdfs:ContainerMembershipProperty[contMP]		rdfs:member[member]
Documentation vocabulary		
rdfs:comment[comment]	rdfs:label[label]	rdfs:seeAlso[seeAlso]
	rdfs:isDefinedBy[isDefined]	

Table 3.1: The RDFS Vocabulary.

- ⟨ex:Sonia rdf:type ex:Employee⟩ asserts that the resource ex:Sonia is an instance of the class ex:Employee;
- ⟨rel:worksWith rdf:type rdf:Property⟩ tells that rel:worksWith is a predicate, i.e., a resource used for labelling edges.

RDFS is expressed as RDF triples using a few more keywords in the rdfs: name space, such as:

- ⟨ex:Employee rdf:type rdfs:Class⟩ indicating that the resource ex:Employee as for type rdfs:Class, it is thus a class.
- ⟨ex:Employee rdfs:subClassOf foaf:Person⟩ meaning that ex:Employee is a subclass of foaf:Person, thus each instance of ex:Employee is also an instance of foaf:Person, like ex:Sonia;
- ⟨ex:worksWith rdfs:range ex:Employee⟩ asserts that any resource used as the extremity of an edge labelled by rel:worksWith is an instance of the ex:Employee class.

The RDFS vocabulary is presented in Table 3.1 as it appears in [HAYES 2004]. The shortcuts that we will use for each of them are given in brackets.

Some terms used in XML/RDF such as rdf:about, rdf:parseType and rdf:resource do not appear in this table as they do not appear in the triple form. They are only used to indicate how to transfer the XML as triples. From now on, we use $RDFSV$ to denote the RDFS vocabulary.

Example 4 (RDFS-graph)**.** *Figure 3.1 displays the RDFS graph with the following assertions:*

```
geo:Grenoble departement geo:Isere
departement rdfs:subPropertyOf is-contained-in
is-contained-in rdfs:range geo:GeographicArea
is-contained-in rdfs:domain geo:GeographicArea
```

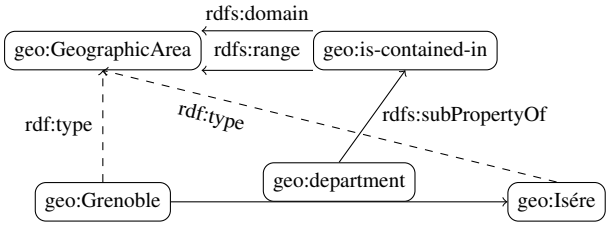


Figure 3.1: An RDFS graph. The arrows in dashed lines are consequences of the RDFS semantics.

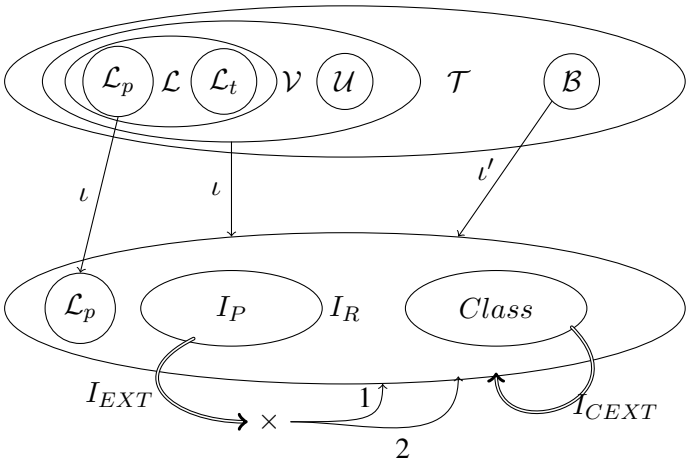


Figure 3.2: Domain structure for RDFS semantics.

Some authors have introduced ρDF [MUÑOZ et al. 2009] as the language corresponding to the RDF and RDFS typing vocabularies and some documents mention "the description logic fragment of RDFS" which is likely to be the same thing. This is on this fragment that we will concentrate, because it is the semantically relevant one.

3.1.2 RDFS semantics

The RDFS semantics is obtained by restricting the set of models of RDFS graphs according to the specific vocabulary of the language. It extends that of RDF by considering some IRIs as identifying classes is a subset of I_R. These are interpreted them in two steps, as properties (see §2.3).

Definition 10 (RDFS interpretation)**.** *An* RDFS interpretation *of a vocabulary V is a tuple $\langle I_R, I_P, Class, I_{EXT}, I_{CEXT}, Lit, \iota\rangle$ such that:*

- *$\langle I_R, I_P, I_{EXT}, \iota\rangle$ is an RDF interpretation;*
- *$Class \subseteq I_R$ is a distinguished subset of I_R identifying if a resource denotes a class of resources;*
- *$I_{CEXT} : Class \to 2^{I_R}$ is a mapping that assigns a set of resources to every resource denoting a class;*
- *$Lit \subseteq I_R$ is the set of literal values, Lit contains all plain literals in $\mathcal{L} \cap V$.*

Specific conditions are added to the resources associated to terms of RDFS vocabularies in an RDFS interpretation to be an RDFS model of an RDFS graph. These conditions include the satisfaction of the RDF and RDF Schema axiomatic triples (Definition 11 and 12) as appearing in the normative semantics of RDF [HAYES 2004].

Definition 11 (RDF axiomatic triples)**.** *RDF axiomatic triples are the triples in the following infinite set:*

$$\langle \texttt{type}, \texttt{type}, \texttt{prop}\rangle \; \langle \texttt{subj}, \texttt{type}, \texttt{prop}\rangle$$
$$\langle \texttt{obj}, \texttt{type}, \texttt{prop}\rangle \; \langle \texttt{pred}, \texttt{type}, \texttt{prop}\rangle$$
$$\langle \texttt{first}, \texttt{type}, \texttt{prop}\rangle \; \langle \texttt{rest}, \texttt{type}, \texttt{prop}\rangle$$
$$\langle \texttt{value}, \texttt{type}, \texttt{prop}\rangle \; \langle \texttt{nil}, \texttt{type}, \texttt{list}\rangle$$
$$\langle \texttt{rdf:\_1}, \texttt{type}, \texttt{prop}\rangle \; \langle \texttt{rdf:\_2}, \texttt{type}, \texttt{prop}\rangle \ldots$$

Definition 12 (RDFS axiomatic triples)**.** *RDFS axiomatic triples are the triples in the following infinite set:*

$$\langle \texttt{type}, \texttt{dom}, \texttt{res}\rangle \; \langle \texttt{type}, \texttt{range}, \texttt{class}\rangle$$
$$\langle \texttt{dom}, \texttt{dom}, \texttt{prop}\rangle \; \langle \texttt{dom}, \texttt{range}, \texttt{class}\rangle$$
$$\langle \texttt{range}, \texttt{dom}, \texttt{prop}\rangle \; \langle \texttt{range}, \texttt{range}, \texttt{class}\rangle$$
$$\langle \texttt{sp}, \texttt{dom}, \texttt{prop}\rangle \; \langle \texttt{sp}, \texttt{range}, \texttt{prop}\rangle$$
$$\langle \texttt{sc}, \texttt{dom}, \texttt{class}\rangle \; \langle \texttt{sc}, \texttt{range}, \texttt{class}\rangle$$
$$\langle \texttt{subj}, \texttt{dom}, \texttt{stat}\rangle \; \langle \texttt{subj}, \texttt{range}, \texttt{res}\rangle$$
$$\langle \texttt{pred}, \texttt{dom}, \texttt{stat}\rangle \; \langle \texttt{pred}, \texttt{range}, \texttt{res}\rangle$$
$$\langle \texttt{obj}, \texttt{dom}, \texttt{stat}\rangle \; \langle \texttt{obj}, \texttt{range}, \texttt{res}\rangle$$
$$\langle \texttt{member}, \texttt{dom}, \texttt{res}\rangle \; \langle \texttt{member}, \texttt{range}, \texttt{res}\rangle$$
$$\langle \texttt{first}, \texttt{dom}, \texttt{list}\rangle \; \langle \texttt{first}, \texttt{range}, \texttt{res}\rangle$$
$$\langle \texttt{rest}, \texttt{dom}, \texttt{list}\rangle \; \langle \texttt{rest}, \texttt{range}, \texttt{list}\rangle$$
$$\langle \texttt{seeAlso}, \texttt{dom}, \texttt{res}\rangle \; \langle \texttt{seeAlso}, \texttt{range}, \texttt{res}\rangle$$
$$\langle \texttt{comment}, \texttt{dom}, \texttt{res}\rangle \; \langle \texttt{comment}, \texttt{range}, \texttt{literal}\rangle$$
$$\langle \texttt{isDefined}, \texttt{dom}, \texttt{res}\rangle \; \langle \texttt{isDefined}, \texttt{range}, \texttt{res}\rangle$$
$$\langle \texttt{label}, \texttt{dom}, \texttt{res}\rangle \; \langle \texttt{label}, \texttt{range}, \texttt{literal}\rangle$$
$$\langle \texttt{value}, \texttt{dom}, \texttt{res}\rangle \; \langle \texttt{value}, \texttt{range}, \texttt{res}\rangle$$

$$\langle \texttt{alt}, \texttt{sc}, \texttt{cont}\rangle \; \langle \texttt{bag}, \texttt{sc}, \texttt{cont}\rangle \qquad \langle \texttt{seq}, \texttt{sc}, \texttt{cont}\rangle$$
$$\langle \texttt{contMP}, \texttt{sc}, \texttt{prop}\rangle$$
$$\langle \texttt{isDefined}, \texttt{sp}, \texttt{seeAlso}\rangle$$
$$\langle \texttt{xmlLit}, \texttt{rdf:type}, \texttt{datatype}\rangle \; \langle \texttt{xmlLit}, \texttt{sc}, \texttt{literal}\rangle$$
$$\langle \texttt{datatype}, \texttt{sc}, \texttt{class}\rangle$$
$$\langle \texttt{rdf:\_1}, \texttt{dom}, \texttt{res}\rangle \; \langle \texttt{rdf:\_1}, \texttt{range}, \texttt{res}\rangle \qquad \langle \texttt{rdf:\_1}, \texttt{rdf:type}, \texttt{contMP}\rangle$$
$$\langle \texttt{rdf:\_2}, \texttt{dom}, \texttt{res}\rangle \; \langle \texttt{rdf:\_2}, \texttt{range}, \texttt{res}\rangle \qquad \langle \texttt{rdf:\_2}, \texttt{rdf:type}, \texttt{contMP}\rangle$$
$$\ldots$$

From this definition, it is clear that any RDFS interpretation is an RDF interpretation and that any RDF interpretation can be extended as different RDFS interpretations: it is sufficient to identify classes.

Definition 13 (RDFS Model)**.** *Let G be an RDFS graph, and $I = \langle I_R, I_P, Class, I_{EXT}, I_{CEXT}, Lit, \iota\rangle$ be an RDFS interpretation of a vocabulary $V \subseteq RDFSV \cup \mathcal{V}$ such that $\mathcal{V}(G) \subseteq V$. Then I is an RDFS model of G if and only if I satisfies the following conditions:*

1. *Simple semantics:*
   a) *there exists an extension ι' of ι to $\mathcal{B}(G)$ such that for each triple $\langle s, p, o\rangle$ of G, $\iota'(p) \in I_P$ and $\langle \iota'(s), \iota'(o)\rangle \in I_{EXT}(\iota'(p))$.*
2. *RDF semantics:*

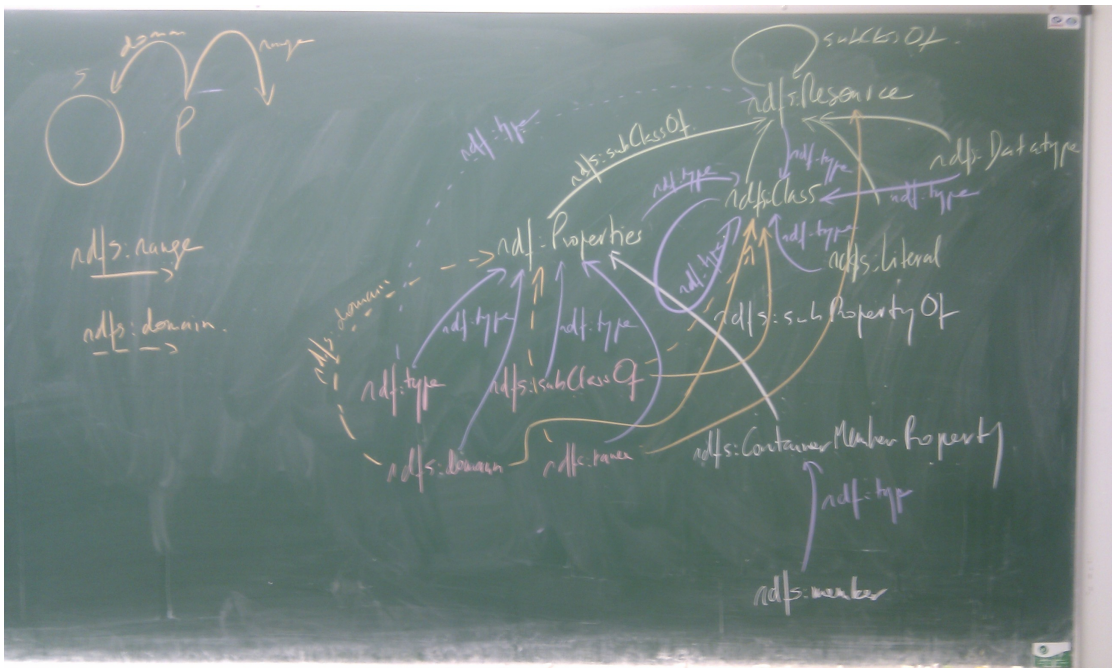


Figure 3.3: Part of the axiomatic triples performed live (exercise: find how many errors are there).

   a) $x \in I_P \Leftrightarrow \langle x, \iota(\texttt{prop})\rangle \in I_{EXT}(\iota(\texttt{type}))$.
   b) *If* $\ell \in term(G)$ *is a typed XML literal with lexical form* w*, then* $\iota'(\ell)$ *is the XML literal value of* w*,* $\iota'(\ell) \in Lit$*, and* $\langle \iota'(\ell), \iota(\texttt{xmlLit})\rangle \in I_{EXT}(\iota(\texttt{type}))$.
   c) I *satisfies all RDF axiomatic triples (Definition 11)*

3. *RDFS Classes:*
   a) $x \in I_R$, $x \in I_{CEXT}(\iota(\texttt{res}))$.
   b) $x \in Class$, $x \in I_{CEXT}(\iota(\texttt{class}))$.
   c) $x \in Lit$, $x \in I_{CEXT}(\iota(\texttt{literal}))$.
4. *RDFS Subproperty:*
   a) $I_{EXT}(\iota(\texttt{sp}))$ *is transitive and reflexive over* I_P.
   b) *if* $\langle x, y\rangle \in I_{EXT}(\iota(\texttt{sp}))$ *then* $x, y \in I_P$ *and* $I_{EXT}(x) \subseteq I_{EXT}(y)$.
5. *RDFS Subclass:*
   a) $I_{EXT}(\iota(\texttt{sc}))$ *is transitive and reflexive over* $Class$.
   b) $\langle x, y\rangle \in I_{EXT}(\iota(\texttt{sc}))$*, then* $x, y \in Class$ *and* $I_{CEXT}(x) \subseteq I_{CEXT}(y)$.
6. *RDFS Typing:*
   a) $x \in I_{CEXT}(y)$, $\langle x, y\rangle \in I_{EXT}(\iota(\texttt{type}))$.
   b) *if* $\langle x, y\rangle \in I_{EXT}(\iota(\texttt{dom}))$ *and* $\langle u, v\rangle \in I_{EXT}(x)$ *then* $u \in I_{CEXT}(y)$.
   c) *if* $\langle x, y\rangle \in I_{EXT}(\iota(\texttt{range}))$ *and* $\langle u, v\rangle \in I_{EXT}(x)$ *then* $v \in I_{CEXT}(y)$.
7. *RDFS Additionals:*
   a) *if* $x \in Class$ *then* $\langle x, \iota(\texttt{res})\rangle \in I_{EXT}(\iota(\texttt{sc}))$.

b) if $x \in I_{CEXT}(\iota(\texttt{datatype}))$ *then* $\langle x, \iota(\texttt{literal})\rangle \in I_{EXT}(\iota(\texttt{sc}))$.

c) if $x \in I_{CEXT}(\iota(\texttt{contMP}))$ *then* $\langle x, \iota(\texttt{member})\rangle \in I_{EXT}(\iota(\texttt{sp}))$.

d) I *satisfies all RDFS axiomatic triples (Definition 12)*

Any RDFS model is an RDF model. It is not true that any RDF model can support an RDFS model because the use of the RDFS vocabulary imposes additional constraints on RDFS models.

Definition 14 (RDFS consequence)**.** *Let* G *and* H *be two RDFS graphs, then* G RDFS-entails H *(denoted by* $G \models_{RDFS} H$*) if and only if every RDFS model of* G *is also an RDFS model of* H.

[HAYES 2004] states that "Since every RDFS interpretation is an RDF interpretation, if G RDFS-entails H then it RDF-entails H". In the errata (and in [HAYES and PATEL-SCHNEIDER 2014]), the converse is given:

Proposition 5. *If* $G \models_{RDF} H$ *then* $G \models_{RDFS} H$.

Proof. $G \models_{RDF} H$ means that $\forall m \in \mathcal{M}_{RDF}(G)$, $\exists \iota'; \forall \langle s, p, o\rangle \in H, \langle \iota'(s), \iota'(o)\rangle \in I_{EXT}(\iota'(p))$, but $\forall m \in \mathcal{M}_{RDFS}(G), m \in \mathcal{M}_{RDF}(G)$ and H does not impose more constraints on its RDFS models than G does since it is entailed by it. Hence, all RDFS-Models of G are RDFS-models of H. □

Even the empty graph RDFS-entails far more assertions than it RDF-entails [HAYES 2004], for example all triples of the form:

```
x rdf:type rdfs:Resource .
```

are true in all RDFS-interpretations of any vocabulary containing the IRI x.

Example 5 (RDFS semantics)**.** *Show why the graph of Example 4 RDFS-entails:*

```
geo:Isere rdf:type geo:GeographicArea
```

The same graph does not Simple RDF entails this assertion because there are RDF-models, which are not RDFS-models and which do not satisfy this triple (exhibit one). Does this holds for `geo:Grenoble` *as well?*

Example 6 (RDFS semantics)**.** *Consider the graph G made of:*

```
ex:Pierre rdf:type ex:TennisPlayer
ex:TennisPlayer rdfs:subClassOf foaf:Person
ex:playsWith rdfs:domain ex:TennisPlayer
ex:defeated rdfs:subPropertyOf ex:playsWith
 _:b ex:defeated ex:Pierre
```

$G \not\models_{RDF}$ `_:b rdf:type foaf:Person` *because there is no subgraph of G that is an instance of this triple.*

But $G \models_{RDFS}$ `_:b rdf:type foaf:Person` *because for any model of G,* $\langle \iota'(\texttt{\_:b}), \iota'(\texttt{ex:Pierre})\rangle \in I_{EXT}(\iota'(\texttt{ex:defeated})$. *By 6c, this implies that* $\iota'(\texttt{\_:b}) \in I_{CEXT}(\iota'(\texttt{ex:TennisPlayer}))$ *In addition,* $\langle \iota'(\texttt{ex:TennisPlayer}), \iota'(\texttt{foaf:Person})\rangle \in I_{EXT}(\iota'(\texttt{rdfs:subClassOf})$ *By 5b, we have that* $I_{CEXT}(\iota'(\texttt{ex:TennisPlayer})) \subseteq I_{CEXT}(\iota'(\texttt{foaf:Person}))$ *Hence,* $\iota'(\texttt{\_:b}) \in I_{CEXT}(\iota'(\texttt{foaf:Person}))$

3.1.3 ter Horst closure

In RDF, it was possible to determine if $G \models_{RDF} H$ by finding an RDF-homorphism between H and G (§2.4). In RDFS, this not so easily possible.

For determining $G \models_{RDFS} H$, a procedure ($\vdash$) have to be designed. In principle, one can consider two approaches to it: A compilation (or data-driven or forward-chaining) procedure which will generate all consequences of G and determine if H belong to them; An evaluation (or query-driven or backward-chaining) procedure which starts with H and search for a proof that it is a consequence of G (see Figure below).

$$G \models_{RDFS} H \quad \equiv \quad G \underset{\text{interpretation}}{\overset{\text{compilation}}{\vdash}} H$$

If the graph (data and ontology) does not change often and queries have to be answered quickly, then the compilation approach is appropriate. If they are constantly evolving and some delay is tolerable for answering queries, then an interpretation approach is more adapted.

Of course, there may be intermediate strategies: for instance, compiling the stable knowledge corresponding to the ontology and interpreting queries with respect to the compiled ontology and the data (supposed less stable).

One possible approach for querying an RDFS graph G in a sound and complete way is by computing the closure graph of G, i.e., the graph obtained by saturating G with all information that can be deduced using a set of predefined rules called RDFS rules, then evaluating the query over the closure graph.

Definition 15 (RDFS closure)**.** *Let G be an RDFS graph on an RDFS vocabulary V. The* RDFS closure *of G, denoted $\hat{G}$, is the smallest set of triple containing G and satisfying the following constraints:*

[RDF1] *all RDF axiomatic triples [*HAYES *2004] are in $\hat{G}$;*

[RDF2] *if $\langle s, p, o\rangle \in \hat{G}$, then $\langle p, \texttt{type}, \texttt{prop}\rangle \in \hat{G}$;*

[RDF3] *if $\langle s, p, \ell\rangle \in \hat{G}$, where ℓ is an* `xmlLit` *typed literal and the lexical representation s is a well-formed XML literal,*
then $\langle s, p, xml(s)\rangle \in \hat{G}$ and $\langle xml(s), \texttt{type}, \texttt{xmlLit}\rangle \in \hat{G}$;

[RDFS 1] *all RDFS axiomatic triples [*HAYES *2004] are in $\hat{G}$;*

[RDFS 6] *if $\langle a, \texttt{dom}, x\rangle \in \hat{G}$ and $\langle u, a, y\rangle \in \hat{G}$, then $\langle u, \texttt{type}, x\rangle \in \hat{G}$;*

[RDFS 7] *if $\langle a, \texttt{range}, x\rangle \in \hat{G}$ and $\langle u, a, v\rangle \in \hat{G}$, then $\langle v, \texttt{type}, x\rangle \in \hat{G}$;*

[RDFS 8a] *if $\langle x, \texttt{type}, \texttt{prop}\rangle \in \hat{G}$, then $\langle x, \texttt{sp}, x\rangle \in \hat{G}$;*

[RDFS 8b] *if $\langle x, \texttt{sp}, y\rangle \in \hat{G}$ and $\langle y, \texttt{sp}, z\rangle \in \hat{G}$, then $\langle x, \texttt{sp}, z\rangle \in \hat{G}$;*

[RDFS 9] *if $\langle a, \texttt{sp}, b\rangle \in \hat{G}$ and $\langle x, a, y\rangle \in \hat{G}$, then $\langle x, b, y\rangle \in \hat{G}$;*

[RDFS 10] *if $\langle x, \texttt{type}, \texttt{class}\rangle \in \hat{G}$, then $\langle x, \texttt{sc}, \texttt{res}\rangle \in \hat{G}$;*

[RDFS 11] *if $\langle u, \texttt{sc}, x\rangle \in \hat{G}$ and $\langle y, \texttt{type}, u\rangle \in \hat{G}$, then $\langle y, \texttt{type}, x\rangle \in \hat{G}$;*

[RDFS 12a] *if $\langle x, \texttt{type}, \texttt{class}\rangle \in \hat{G}$, then $\langle x, \texttt{sc}, x\rangle \in \hat{G}$;*

[RDFS 12b] *if $\langle x, \texttt{sc}, y\rangle \in \hat{G}$ and $\langle y, \texttt{sc}, z\rangle \in \hat{G}$, then $\langle x, \texttt{sc}, z\rangle \in \hat{G}$;*

[RDFS 13] *if $\langle x, \texttt{type}, \texttt{contMP}\rangle \in \hat{G}$, then $\langle x, \texttt{sp}, \texttt{member}\rangle \in \hat{G}$;*

[RDFS 14] *if $\langle x, \texttt{type}, \texttt{datatype}\rangle \in \hat{G}$, then $\langle x, \texttt{sc}, \texttt{literal}\rangle \in \hat{G}$.*

It is easy to show that this closure always exists and can be obtained by turning the constraints into rules, thus defining a closure operation.

Example 7 (RDFS Closure)**.** *The RDFS closure of the RDF graph of Example 2 (p. 19) augmented by the RDFS triples of Example 24 (p. 90) contains, in particular, the following assertions:*

```
dm:bcd rn:inhibits dm:cad. // [RDFS 9]
dm:hb rn:regulates dm:kni. // [RDFS 9]
dm:hb type rn:gene.        // [RDFS 6]
```

Because of axiomatic triples, this closure may be infinite, but a finite and polynomial closure, called *partial closure*, has been proposed independently in [BAGET 2003] and [TER HORST 2005].

Definition 16 (Partial RDFS closure)**.** *Let G and H be two RDFS graphs on an RDFS vocabulary V, the* partial RDFS closure *of G given H, denoted $\hat{G}\backslash H$, is obtained in the following way:*

1. *let k be the maximum of i's such that `rdf:_i` is a term of G or of H;*
2. *replace the rule [RDF 1] by the rule*
   [RDF 1P] add all RDF axiomatic triples [HAYES 2004] except those that use `rdf:_i` with $i > k$;
   In the same way, replace the rule [RDFS 1] by the rule
   [RDFS 1P] add all RDFS axiomatic triples except those that use `rdf:_i` with $i > k$;
3. *apply the modified rules.*

Applying the partial closure to an RDFS graph permits to reduce RDFS entailment to simple RDF entailment.

Proposition 6 (Completeness of partial RDFS closure [HAYES 2004; TER HORST 2005])**.** *Let G be a satisfiable RDFS graph and H an RDFS graph, then $G \models_{RDFS} H$ if and only if $(\hat{G}\backslash H) \models_{RDF} H$.*

Incomplete proof. The proof has to rely on two arguments: (1) that $\hat{G} \equiv_{RDFS} G$, this is left as an exercise to show that each rule is correct w.r.t. Definition 13, hence $G \models_{RDFS} H \Leftrightarrow \hat{G} \models_{RDFS} H$; (2) that $\hat{G}$ is actually maximal, i.e., $\forall t \in \hat{G} \setminus G, G \models t$, and more tricky that $\hat{G}\backslash H$ is, i.e., that $\forall t \in H; G \models t \Rightarrow t \in \hat{G}\backslash H$, this entails $\hat{G} \models_{RDFS} H \Leftrightarrow \hat{G}\backslash H \models_{RDF} H$. The *only* part is easy, each triple $t \in H$ is RDF-entailed by $\hat{G}\backslash H$, then it is RDFS-entailed by G, thanks to Proposition 5. The *if* part must apply to all models and hence is more elaborate and given in [TER HORST 2005]. □

$\hat{G}\backslash H$ is limited to some consequences that may affect H, hence this completeness is relative to H. The completeness does not hold if G is not satisfiable because in such a case, any graph H is a consequence of G and $\models_{RDF}$ does not reflect this (no RDF graph can be inconsistent). An RDFS graph can be unsatisfiable only if it contains datatype conflicts [TER HORST 2005] which can be found in polynomial time.

This approach is intermediate since it compiles the graph G with respect to H and then it evaluates H with respect to $\hat{G}\backslash H$ (by looking for an RDF-homomorphism). This seems to defeat the purpose of compilation since it has to be recomputed for every query. However, this compilation is a full compilation up to the highest n, then it will be usable for any graph H with a lower such n and if it has a greater n, then it is possible to extend the previous compilation.

Example 8. *Consider the graph made of the following triples:*

```
a:Book rdfs:subClassOf a:Document .
a:writtenBy rdfs:subPropertyOf a:createdBy .
a:writtenBy rdfs:range a:Writer .
a:createdBy rdfs:range a:Creator .
a:Creator rdfs:subClassOf a:Person .
b:WMLForDummies a:writtenBy b:Pierre .
```

Does it RDF entails any of the triples:

```
?x rdf:type a:Book . // there exists a book
?x rdf:type a:Person . // there exists a person
a:Writer rdfs:subClassOf a:Creator . // writers are creators
b:Pierre rdf:type a:Person . // Pierre is a person
```

Compute the closure of the initial set of triples. Which of the four triples is RDFS-entailed?

3.1.4 Computational properties

We can define the RDFS ENTAILMENT problem in the same way as we defined SIMPLE RDF ENTAILMENT:

RDFS ENTAILMENT
Instance: two RDFS graphs G and H.
Question: Does $G \models_{\text{RDFS}} H$?

RDFS ENTAILMENT is an NP-complete problem [GUTIERREZ et al. 2004].

3.1.5 Conclusion

Full RDF extends simple RDF with a specific vocabulary. Similarly, RDF Schema extends RDF with another vocabulary. These vocabularies express constraints on the entities in RDF graphs: their type and structure for RDF and further constraints on domains of properties and class/property specialisation. These constraints further extend the definition of RDFS interpretations, and reduces the interpretations which are models of RDF and RDFS graphs. Hence the notion of consequence is different with Full RDF or RDFS graph.

It is not possible anymore to reduce entailment to the existence of a homomorphism. However, by computing a closure of the entailing graph, it is possible to generate a larger graph against which RDFS entailment correspond to simple RDF entailment (and thus homomorphism testing is again complete and correct). Although, the closure may be infinite, it is possible to compute a smaller, finite useful subset called the partial closure against which homomorphism checking is again complete and correct.

3.2 The web ontology language OWL

RDF and RDFS allows to assert relations between classes and properties, e.g., subClassOf, but they do not allow to construct these from the inside. The OWL language [HORROCKS, PATEL-SCHNEIDER, and VAN HARMELEN 2003; DEAN and SCHREIBER 2004] is dedicated to class and property definitions. Inspired from description logics [BAADER et al. 2003], it provides constructors to constrain them precisely. The W3C has released a new version of OWL (OWL 2.0) [BECKETT 2009].

We present below the syntax, semantics and the different standard sublanguages of OWL.

3.2.1 OWL syntax

The OWL syntax, based on RDF, introduces a specific vocabulary in the `owl:` name space. But, OWL is more than a vocabulary: there are many RDF graphs with OWL vocabulary which cannot be interpreted in OWL. Hence, not all RDF graphs using this vocabulary are necessarily valid OWL ontologies: further constraints have to be satisfied.

It is more reasonable to think that OWL has a syntax, cast in RDF syntax, in the more classic sense of the term. OWL 2 specifies five different syntaxes [BECKETT 2009]:

XML/RDF is the only mandatory exchange syntax, it is based on XML with a very regular, but verbose, structure;

OWL/XML an ugly but straightforward XML syntax;

Functional syntax as its name indicates, this syntax expresses the abstract trees of the OWL concepts and is better suited for specifying its semantics (and manipulating it inductively);

Manchester syntax is supposedly a “user-friendly” syntax, using syntactic sugar;

Turtle itself an extension of the n3 syntax for RDF, it is also supposed to be more readable by users.

It is clear that, in spite of their names, the import of these languages is not their syntax, but their structure and semantics. We will mostly use the XML/RDF and Turtle syntaxes.

In particular, OWL abstractly defines *class descriptions* and *property descriptions*. These can be defined recursively like terms

Definition 17 (OWL terminology)**.** *In OWL, the terminology is still divided among literals $\mathcal{L}$, IRIs $\mathcal{U}$ and variables $\mathcal{B}$. However, the set $\mathcal{U}$ is divided into many sets ($\langle \mathcal{C}, \mathcal{D}, \mathcal{I}, \mathcal{P}_d, \mathcal{P}_i \rangle$):*

$\mathcal{C}$ *the set of class names containing* `owl:Thing` *and* `owl:Nothing`*;*
$\mathcal{D}$ *the set of data types containing* `rdfs:Literal`*;*
$\mathcal{I}$ *the set of individual names;*
$\mathcal{P}_d$ *the set of data type property names;*
$\mathcal{P}_i$ *the set of object property names;*

In the initial definition, there were also others categories:

$\mathcal{P}_a = V_{AP}$ the set of annotation property names containing `rdfs:label`, `rdfs:comment`, `rdfs:seeAlso`, `rdfs:versionInfo` and `rdfs:isDefinedBy`. These have been renamed *facets* in OWL 2;
$\mathcal{O} = V_O$ the set of ontology names. They have been dropped in OWL 2 semantics.

but, in the following, we will only consider $\mathcal{C}$, $\mathcal{D}$, $\mathcal{I}$, $\mathcal{P}_d$, $\mathcal{P}_i$ since these are the only ones with a meaningful semantics. We will also ignore documentation vocabulary.

The OWL vocabulary is given in Table 3.2 following [DEAN and SCHREIBER 2004] for OWL and [BAO et al. 2009] for OWL 2. We only cover the main terms, miscellaneous one are not considered here. One of the main difference between OWL and OWL 2 is a uniform treatment of datatypes which is also lightly covered here.

The intuitive semantics for the main OWL constructors is as follows:

- RDF keywords (`rdf:type`, `rdf:Property`) and RDFS' (`rdfs:subClassOf`, `rdfs:subPropertyOf`, `rdfs:range`, `rdfs:domain`) are used with the same semantics.
- `owl:Class` is a new (meta)class.
- `owl:sameAs` and `owl:differentFrom` are used to assert that two resources are equal or different.
- `owl:inverseOf` asserts that a property p is the converse of a property p' (in this case, the triple $\langle s\ p\ o \rangle$ entails $\langle o\ p'\ s \rangle$); other characteristics may be assigned to properties such as reflexivity (`owl:ReflexiveProperty`), transitivity (`owl:TransitiveProperty`), symmetry (`owl:SymmetricProperty`) or functionality (`owl:FunctionalProperty`).
- `owl:allValuesFrom` associates a class c to a relation p. This defines the class of objects x such that if $\langle x\ p\ y \rangle$ holds, then y belong to c (this is a universally quantified role in description logics). `owl:someValuesFrom` encodes existentially quantified roles.
- `owl:minCardinality` (resp. `owl:maxCardinality`) allows one to define the class of objects related to at least (resp. at most) a specific number of objects through a given property. A qualified version of these constructors constrains, in addition, that these objects belong to a particular class.
- `owl:oneOf` defines a class in comprehension by enumerating the set of its instances.
- `owl:hasValue` constrains a property to have a particular individual as value.
- `owl:disjointWith` asserts that two classes cannot have a common instance.
- `owl:unionOf`, `owl:intersectionOf` and `owl:complementOf` define a class as the disjunction, the conjunction or the negation of other classes.
- `owl:hasSelf` defines the class of objects related to themselves through a specific relation.
- `owl:hasKey` asserts that a set of properties is a key for a class, i.e., that two distinct instances cannot share the same values for these properties.

	Typing vocabulary	
owl:Class	owl:Thing	owl:Nothing
rdf:Property	owl:Data(type)Property	owl:ObjectProperty
owl:topObjectProperty[2]	owl:bottomObjectProperty[2]	owl:topDataProperty[2]
owl:bottomDataProperty[2]		
	Class/Datatype$^+$ constructor	
owl:intersectionOf$^+$	owl:unionOf$^+$	owl:complementOf
owl:oneOf$^+$	owl:datatypeComplementOf[2]	owl:disjointUnionOf[2?]
	Property restrictions	
owl:Restriction	owl:onProperty	owl:onClass[2]
owl:someValuesFrom$^+$	owl:minQualifiedCardinality[2]	owl:minCardinality
owl:allValuesFrom$^+$	owl:maxQualifiedCardinality[2]	owl:maxCardinality
owl:hasValue	owl:qualifiedCardinality[2]	owl:cardinality
	Class constraints	
owl:equivalentClass	rdfs:subClassOf	owl:disjointWith
owl:hasSelf[2]	owl:hasKey[2]	
	Property constraints	
owl:equivalentProperty	rdfs:subPropertyOf	owl:inverseOf
	owl:propertyChainAxiom	
owl:FunctionalProperty	owl:TransitiveProperty	owl:SymmetricProperty
owl:AsymmetricProperty[2]	owl:ReflexiveProperty[2]	owl:IreflexiveProperty[2]
rdfs:domain	owl:InverseFunctionalProperty	rdfs:range
	Individual constraints	
	owl:sameAs	owl:differentFrom
	Global constraints	
owl:AllDifferent	owl:AllDisjointpropertiess[2]	owl:AllDisjointClasses[2]
	owl:Ontology	owl:imports
	Documentation	
owl:versionInfo	owl:priorVersion	
owl:DeprecatedClass	owl:backwardCompatibleWith	
owl:DeprecatedProperty	owl:AnnotationProperty	owl:OntologyProperty

Table 3.2: The OWL vocabulary. OWL 2 primitives are followed by an ”2” exponent; primitives applicable to datatypes since OWL 2 are marked with a ”+” exponent.

– `owl:propertyChainAxiom` composes several relations and properties to obtain a new relation or property.

We have not mentioned all constructors. Many of them can be trivially implemented by using the cited ones, e.g., `owl:equivalentClass` asserting that two classes are equivalent can be expressed with two `rdfs:subClassOf` assertions. OWL also uses data types that are not considered here. They are however important as they can lead to inconsistency.

Example 9. *The following example:*

$$ChemistryProfessor \sqsubseteq Professor \sqcap \forall teaches.ChemistryLecture$$
$$teaches = taughtBy^{-1}$$
$$\top \sqsubseteq \forall teaches.Lecture \sqcap \forall teaches^{-1}.Professor$$

can be expressed in OWL by:

```
<owl:Class rdf:about="#ChemistryProfessor">
  <rdfs:subClassOf>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#Professor" />
        <owl:Restriction>
          <owl:onProperty rdf:resource="#teaches" />
          <owl:allValuesFrom rdf:resource="#ChemistryLecture" />
        </owl:Restriction>
      </owl:intersectionOf>
     </owl:Class>
  </rdfs:subClassOf>
</owl:Class>

<owl:ObjectProperty rdf:about="#teaches">
  <owl:inverseOf rdf:resource="#taughtBy" />
  <rdfs:domain rdf:resource="#Professor" />
  <rdfs:range rdf:resource="#Lecture" />
</owl:ObjectProperty>
```

It is noteworthy that the last statements are an encoding of `rdfs:domain` *and* `rdfs:range`*. In particular, the semantics of OWL is defined so that this statement indicates that:*

$$I(\texttt{teaches}) = I(\texttt{taughtBy})^{-1} \cap (I(\texttt{Professor}) \times D) \cap (D \times I(\texttt{Lecture}))$$

The same can be expressed in the terrible syntax of triples:

```
ex:ChemistryProfessor rdf:type owl:Class
ex:ChemistryProfessor rdfs:subClassOf _:b1
_:b1 rdf:type owl:Class
_:b1 owl:intersectionOf _:b2
_:b2 rdf:type rdf:List
_:b2 rdf:_1 ex:Professor
ex:Professor rdf:type owl:Class
_:b2 rdf:_2 _:b3
_:b3 rdf:type owl:Restriction
_:b3 owl:onProperty ex:teaches
_:b3 owl:allValuesFrom ex:ChemistryLecture
```

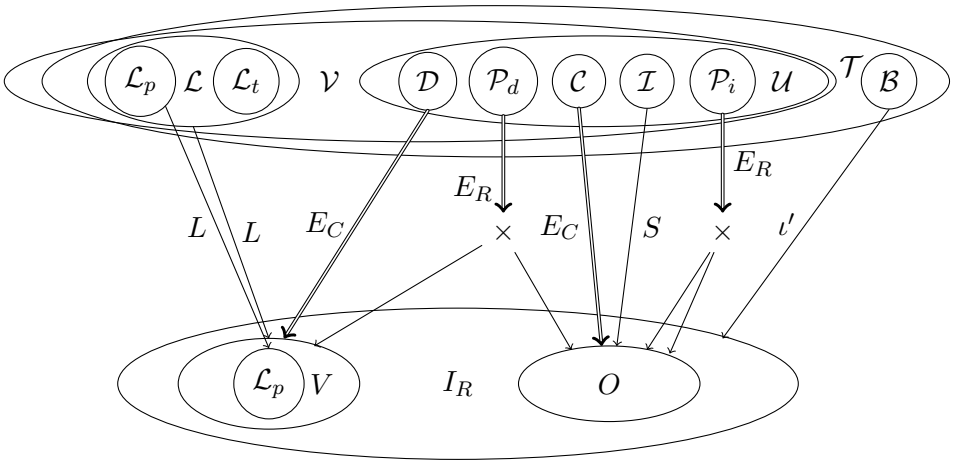


Figure 3.4: Domain structure for OWL semantics.

```
ex:teaches rdf:type owl:ObjectProperty.
ex:teaches owl:inverseOf ex:taughtBy
ex:teaches rdfs:domain ex:Professor
ex:teaches rdfs:range ex:Lecture
```

3.2.2 OWL semantics

The semantics of OWL constructs is given in [PATEL-SCHNEIDER et al. 2004; MOTIK, PATEL-SCHNEIDER, et al. 2009] following a description logic style.

Definition 18 (OWL interpretation)**.** *An OWL interpretation over a vocabulary $\langle \mathcal{L}, \mathcal{C}, \mathcal{D}, \mathcal{I}, \mathcal{P}_d, \mathcal{P}_i \rangle$ is a tuple $I = \langle I_R, E_C, E_R, L, S, V, O \rangle$ such that:*

- *I_R (resources) $\neq \varnothing$;*
- *$O \neq \varnothing$;*
- *$O \subseteq I_R$;*
- *$O \cap V = \varnothing$;*
- *$V \subseteq I_R$ (plus extras)*
- *$E_C : \mathcal{C} \to 2^O$;*
- *$E_C : \mathcal{D} \to 2^V$;*
- *$E_R : \mathcal{P}_d \to 2^{O \times V}$;*
- *$E_R : \mathcal{P}_i \to 2^{O \times O}$;*
- *$L : \mathcal{L} \to V$ (plus extra):*
- *$S : \mathcal{C} \cup \mathcal{D} \cup \mathcal{I} \cup \mathcal{P}_d \cup \mathcal{P}_i \to I_R$ (plus extra)*
- *$S(\mathcal{I}) \subseteq O$;*
- *more about datatypes*

Compound expressions are interpreted according the the classical description logic semantics.

Definition 19 (Interpretation of compound expressions)**.** *Let $I = \langle I_R, E_C, E_R, L, S, V, O \rangle$ be an OWL interpretation over a vocabulary $\langle \mathcal{L}, \mathcal{C}, \mathcal{D}, \mathcal{I}, \mathcal{P}_d, \mathcal{P}_i \rangle$ the interpretation of compound class expressions by I*

is defined by (all keywords are in the OWL namespace):

$$
\begin{aligned}
E_C(\texttt{owl:Thing}) &= O \\
E_C(\texttt{owl:Nothing}) &= \varnothing \\
E_C(\texttt{rdfs:Literal}) &= V \\
E_C(\texttt{complementOf}(c)) &= O - E_C(c) \\
E_C(\texttt{unionOf}(c, c')) &= E_C(c) \cup E_C(c') \\
E_C(\texttt{intersectionOf}(c, c')) &= E_C(c) \cap E_C(c') \\
E_C(\texttt{oneOf}(i_1, \dots i_n)) &= \{S(i_1), \dots S(i_n)\} \\
E_C(\texttt{oneOf}(v_1, \dots v_n)) &= \{L(i_1), \dots L(i_n)\} \\
E_C(\texttt{restriction}(p, \texttt{allValuesFrom}(r))) &= \{x \in O; \forall \langle x, y\rangle \in E_R(p), y \in E_C(r)\} \\
E_C(\texttt{restriction}(p, \texttt{someValueFrom}(r))) &= \{x \in O; \exists \langle x, y\rangle \in E_R(p) \wedge y \in E_C(r)\} \\
E_C(\texttt{restriction}(p, \texttt{hasValue}(v))) &= \{x \in O; \langle x, L(v)\rangle \in E_R(p)\} \\
E_C(\texttt{restriction}(p, \texttt{hasValue}(i))) &= \{x \in O; \langle x, S(i)\rangle \in E_R(p)\} \\
E_C(\texttt{restriction}(p, \texttt{minCardinality}(n))) &= \{x \in O; |\{\langle x, y\rangle \in E_R(p)\}| \geq n\} \\
E_C(\texttt{restriction}(p, \texttt{maxCardinality}(n))) &= \{x \in O; |\{\langle x, y\rangle \in E_R(p)\}| \leq n\} \\
E_C(\texttt{restriction}(p, \texttt{cardinality}(n))) &= \{x \in O; |\{\langle x, y\rangle \in E_R(p)\}| = n\} \\
E_C(\texttt{restriction}(p, \texttt{minQCardinality}(n, C))) &= \{x \in O; |\{\langle x, y\rangle \in E_R(p); y \in E_C(C)\}| \geq n\} \\
E_C(\texttt{restriction}(p, \texttt{maxQCardinality}(n, C))) &= \{x \in O; |\{\langle x, y\rangle \in E_R(p); y \in E_C(C)\}| \leq n\} \\
E_C(\texttt{restriction}(p, \texttt{qCardinality}(n, C))) &= \{x \in O; |\{\langle x, y\rangle \in E_R(p); y \in E_C(C)\}| = n\} \\
E_R(\texttt{inverseOf}(p)) &= \{\langle x, y\rangle \in O \times O; \langle y, x\rangle \in E_R(p)\} \\
E_R(\texttt{propertyChainAxiom}(p_1, \dots p_n)) &= \{\langle y_0, y_n\rangle \in O \times O; \exists y_0, \dots y_n; \\
&\qquad \forall i \in [1, n] \langle y_{i-1}, y_i\rangle \in E_R(p_i)\}
\end{aligned}
$$

Like for description logics, it would be possible to interpret other type of compound expressions (like property path expressions).

Definition 20 (Axiom satisfaction)**.** *Let $I = \langle I_R, E_C, E_R, L, S, V, O\rangle$ be an OWL interpretation over a vocabulary $\langle \mathcal{L}, \mathcal{C}, \mathcal{D}, \mathcal{I}, \mathcal{P}_d, \mathcal{P}_i\rangle$ an axiom is said to be satisfied (all keywords are in the OWL namespace):*

$$
\begin{aligned}
\texttt{equivalentClass}(c, c') &\text{ iff } E_C(c) = E_C(c') \\
\texttt{subClassOf}(c, c') &\text{ iff } E_C(c) \subseteq E_C(c') \\
\texttt{disjointWith}(c, c') &\text{ iff } E_C(c) \cap E_C(c') = \varnothing \\
\texttt{equivalentProperty}(p, p') &\text{ iff } E_R(p) = E_R(p') \\
\texttt{subPropertyOf}(p, p') &\text{ iff } E_R(p) \subseteq E_R(p') \\
\texttt{sameAs}(i, i') &\text{ iff } S(i) = S(i') \\
\texttt{differentFrom}(i, i') &\text{ iff } S(i) \neq S(i') \\
\texttt{AllDifferent}(i_0, \dots i_n) &\text{ iff } \forall j, k \in [0\ n], j \neq k \Rightarrow S(i) \neq S(i') \\
\texttt{SymmetricProperty}(p) &\text{ iff } \forall \langle x, y\rangle \in E_R(p), \langle y, x\rangle \in E_R(p) \\
\texttt{TransitiveProperty}(p) &\text{ iff } \forall \langle x, y\rangle, \langle y, z\rangle \in E_R(p), \langle x, z\rangle \in E_R(p) \\
\texttt{FunctionalProperty}(p) &\text{ iff } \forall \langle x, y\rangle, \langle x, z\rangle \in E_R(p), y = z \\
\texttt{InverseFunctionalProperty}(p) &\text{ iff } \forall \langle y, x\rangle, \langle z, x\rangle \in E_R(p), y = z
\end{aligned}
$$

A model is defined as usual as an interpretation that satisfies all axioms. It also satisfies the constraints raised by the category of each term.

Definition 21 (OWL model)**.** *An OWL interpretation $I = \langle I_R, E_C, E_R, L, S, V, O\rangle$ over a vocabulary $\langle \mathcal{L}, \mathcal{C}, \mathcal{D}, \mathcal{I}, \mathcal{P}_d, \mathcal{P}_i\rangle$ is a model of an OWL Ontology $\mathcal{O}$ iff:*

- *each IRI used in class (resp. datatype property, object property, individual, datatype) position in $\mathcal{O}$, belongs to $\mathcal{C}$ (resp. $\mathcal{P}_d$, $\mathcal{P}_i$, $\mathcal{I}$, $\mathcal{D}$);*
- *each literal in $\mathcal{O}$ belongs to $\mathcal{L}$;*
- *there exists an extension J of I to variables (blanks) such that J satisfies all axioms in $\mathcal{O}$.*

Consequence, inconsistence, etc. are defined as usual.

Example 10. *Here is an example of an OWL ontology in the terrible syntax of a set of triples:*

```
ex:Grenoble rdf:type geo:City
geo:City rdfs:subClassOf geo:GeographicArea
ex:is-contained-in  owl:inverseOf ex:contains
geo:City rdfs:subClassOf  _:x
_:x rdf:type  owl:Restriction
_:x owl:onProperty  ex:is-contained-in
_:x owl:maxCardinality  "1"^^xsd:Integer
geo:City rdfs:subClassOf  _:y
_:y rdf:type  owl:Restriction
_:y owl:onProperty  ex:country
_:y owl:someValuesFrom geo:Country
```

It entails the following sets of triples:

```
ex:Grenoble rdf:type  geo:GeographicArea
ex:Isere ex:contains  ex:Grenoble
ex:Grenoble ex:country  _:z
_:z rdf:type ex:Country
```

Moreover, if one adds:

```
ex:Grenoble ex:country  ex:France
ex:country rdfs:subPropertyOf ex:is-contained-in
```

this entails:

```
ex:France owl:sameAs ex:Isere
```

which is obviously wrong. So what is the problem?

Example 11. *Consider the ontology made of the following statements:*

$$ChemistryProfessor \sqsubseteq ScienceProfessor \sqcap \forall teaches.ChemistryLecture$$
$$ScienceProfessor \sqsubseteq Professor \sqcap \exists teaches.ScienceLecture$$

The OWL semantics makes that: $\models_{OWL} ChemistryProfessor \sqsubseteq Professor$ *holds because for any model,*

$$\begin{aligned}
E_C(ChemistryProfessor) &\subseteq E_C(ScienceProfessor \sqcap \forall teaches.ChemistryLecture) \\
&= E_C(ScienceProfessor) \cap E_C(\forall teaches.ChemistryLecture) \\
&= E_C(ScienceProfessor) \\
&\cap \{o | \forall y; \langle o, y\rangle \in E_R(teaches), y \in E_C(ChemistryLecture)\} \\
E_C(ScienceProfessor) &\subseteq E_C(Professor \sqcap \exists teaches.ScienceLecture) \\
&= E_C(Professor) \cap E_C(\exists teaches.ScienceLecture) \\
&= E_C(Professor) \\
&\cap \{o | \exists y; \langle o, y\rangle \in E_R(teaches), y \in E_C(ChemistryLecture)\}
\end{aligned}$$

Hence, $E_C(ChemistryProfessor) \subseteq E_C(ScienceProfessor) \subseteq E_C(Professor)$ *is true in all models.*

It is not true that $\models_{OWL}$ `_:b rdf:type ex:ChemistryLecture`. *However, if one adds the statement that* `ex:Paul rdf:type ex:ChemistryProfessor`, *then* $\models_{OWL}$ `_:b rdf:type ex:ChemistryLecture`.

3.2.3 Tableau method for OWL

OWL in general also enjoys infinite closure (for instance, a maximum cardinality restriction entails maximum cardinality restrictions of all greater integers), but the patterns of entailed statements are wider. Hence, it is not advised to use closure.

Inference in OWL is usually provided by dedicated description logic provers based on the tableau method, resolution or other transformations. The tableau method (or semantic tableau method) is a method which attempts at creating a model for a set of formulas [BAADER et al. 2003]. For that purpose, it explores the space of possible models using rules which guarantee that the exploration is exhaustive and complies with the semantics of the language.

In description logics, tableau reasoning usually proceeds by refutation. In order to prove that $G \models_{OWL} H$, instead of investigating all models of G for checking that H holds in all of them, one starts with G and an encoding of the negation of H and tries to build *one* model satisfying them ($G \wedge \neg H$). If such a model is found this means that there exist a model of G not satisfying H, so H is not entailed by G. If no such model is found, then this means that H holds in models of G and is thus entailed by G.

For this to be possible, it is necessary that the logic be negation complete, i.e., that the negation of any formula of the language can be expressed by formula of the language.

A tableau procedure has been designed for most of the OWL language. We provide below an example of tableau for the $\mathcal{ALC}$ language which is a small subset of OWL (see [HORROCKS and SATTLER 2007] for a fuller treatment). An $\mathcal{ALC}$ TBox is a set of general concept inclusion axioms of the form $C \sqsubseteq C'$. Concepts are defined by:

$$C = A | \bot | \top | C \sqcap C' | C \sqcup C' | \neg C | \forall R.C | \exists R.C$$

and roles are simply atomic roles ($R = r$).

The tableaux method for $\mathcal{ALC}$ separates the properly ontological content of the formula —roughly, the OWL part— as a TBox T and the assertional content —the RDF part— as an ABox A. Moreover, the ABox have to be expressed into negation normal form, i.e., negations only apply to atomic classes. It proceeds by applying rules for building a model. Such models can be thought of as RDF graphs: they are labelled graphs with edges labelled by roles and nodes x labelled by the set of class expressions to which the node belongs ($L(x)$). These rules develop a tree of the possible models. Some specific rules, called clashes

$\sqcap$-rule
Condition: $C \sqcap D \in L(x)$, x is not blocked; $\{C, D\} \not\subseteq L(x)$
Action: $L(x) := L(x) \cup \{C, D\}$

$\sqcup$-rule
Condition: $C \sqcup D \in L(x)$, x is not blocked; $C \notin L(x)$, $D \notin L(x)$
Action: $L(x) := L(x) \cup \{C\}$, or $L(x) := L(x) \cup \{D\}$

$\exists$-rule
Condition: $\exists r.C \in L(x)$, x is not blocked; $\nexists y; r(x, y) \wedge C \in L(y)$
Action: create a new node y with $L(\langle x, y\rangle) = \{r\}$ and $L(y) = \{C\}$

$\forall$-rule
Condition: $\forall r.C \in L(x)$, x is not blocked; $\exists y; r(x, y) \wedge C \notin L(y)$
Action: $L(y) := L(y) \cup \{C\}$

$\sqsubseteq$-rule
Condition: $C \sqsubseteq D \in T$, x is not blocked, $\neg C \sqcup D \notin L(x)$
Action: $L(x) := L(x) \cup \{\neg C \sqcup D\}$

$\neg$**-clash** : $\exists x; \{C, \neg C\} \subseteq L(x)$
$\bot$**-clash** : $\exists x; \bot \in L(x)$

Table 3.3: Completion rules and Clash condition for $\mathcal{ALC}$.

denote an impossible model and hence close the branch of the tree. If such a clash is satisfied, then the current representation cannot be turned into a model and the algorithm must explore eventual alternative representations. A specific technique, called blocking, is used to avoid generating infinite branches due to infinitely expanding rules. Rules and clashes for $\mathcal{ALC}$ are provided in Table 3.3. Rules are applied until none is applicable. If there is a branch which does not contain a clash, then a model has been built.

3.2.4 OWL 2 and profiles

The initial OWL specification defined three different sublanguages (OWL Full, OWL DL and OWL Lite). Fortunately people realised that the framework of description logics was rich enough to accommodate many more languages. We consider here the state of the OWL specifications. In addition, of numerous minor changes, OWL 2 introduces the notion of profiles [MOTIK, CUENCA GRAU, et al. 2009], reminiscent of description logic modularity which describe a particular subset of OWL. These profiles cover the previous OWL sublanguages and introduce three new OWL 2 profiles. A profile is characterised by the vocabulary it uses and the syntactic form of the graphs it accepts.

These are:

OWL Lite
- has separate domains of discourse for properties and classes;
- contains all RDF constructors, i.e., it allows to declare an individual as member of a class and to relate individuals and data values through properties;
- uses part of the RDFS vocabulary (rdfs:subClassOf, rdfs:subPropertyOf, rdfs:range, rdfs:domain), with the same semantics;
- enables the definition of new classes (`owl:Class`) as more specific or equivalent to the conjunction of other classes;

- `owl:sameIndivid ualAs` and `owl:differentIndividualFrom` assert that two individuals are equivalent or different;
- properties characteristics can be asserted through: `owl:inverseOf` states that a property p is the converse of another p' (hence, the triple $\langle s\ p\ o\rangle$ entails $\langle o\ p'\ s\rangle$); other characteristics are transitivity (`owl:TransitiveProperty`), functionality (`owl:FunctionnalProperty`), symmetry (`owl:SymmetricProperty`) or inverse functionality (`owl:InverseFunctionalProperty`),
- `owl:allValuesFrom` defines the class of objects for which all values to property p belong to class c;`owl:someValuesFrom` defines the class of objects for which there exists a value to property p belonging to class c;
- `owl:minCardinality` (resp. `owl:maxCardinality`) defines the class of objects which have at least (resp. at most) n values for property p. In OWL Lite, n can only be 0 or 1.

OWL Lite corresponds to the $\mathcal{SHIF}(\mathcal{D})$ description logic (concept conjunction, disjunction and negation+universal role restriction+qualified existential role restriction+transitive roles+role hierarchy+inverse roles+functional roles+datatypes). $\mathcal{SHIF}(\mathcal{D})$ entailment can be decided in EXPTIME [HORROCKS, PATEL-SCHNEIDER, and VAN HARMELEN 2003] and there are efficient algorithms for solving the problem.

OWL 2 RL is a fragment of OWL Full that can be implemented with a rule language.

OWL 2 QL corresponds roughly to DL-Lite (§3.2.5);

OWL DL contains all constructors, with particular constraints on their use that warrants the decidability of entailment tests. So, OWL DL:

- Contains all OWL Lite constructors;
- Allows any positive integer in cardinality constraints but prohibits them on transitive properties or their inverse or any of their subproperties;
- `owl:oneOf` defines a class through the set of its instances,
- `owl:hasValue` defines the class of objects having only a particular value v for some property p;
- `owl:disjointWith` asserts that two classes have no common instances;
- `owl:unionOf` and `owl:complementOf` allow the definition of a class a the union of two classes or the complement of one class.

OWL DL corresponds to the $\mathcal{SHOIN}(\mathcal{D})$ description logic ($\mathcal{SHIF}(\mathcal{D})$+nominals). $\mathcal{SHOIN}(\mathcal{D})$ entailment test has a non deterministic exponential time complexity (NEXPTIME) [HORROCKS, PATEL-SCHNEIDER, and VAN HARMELEN 2003]. So far there is no known complete algorithm for this problem that can be implemented.

OWL 2 EL retains an expressive power in the definition of classes and properties restricted to existential quantification (it drops universal quantification, cardinality restrictions, disjunction, negation and relation properties);

OWL Full uses all constructors without any constraint. It does not corresponds to a well identified class of description logics due to its non standard semantics. Since it contains negation and definition of sets which can contain defined sets, it is most likely undecidable.

- Contains all OWL DL constructors;
- Contains all RDF Schema constructors;
- Allows to use classes in place of individuals in constructors.

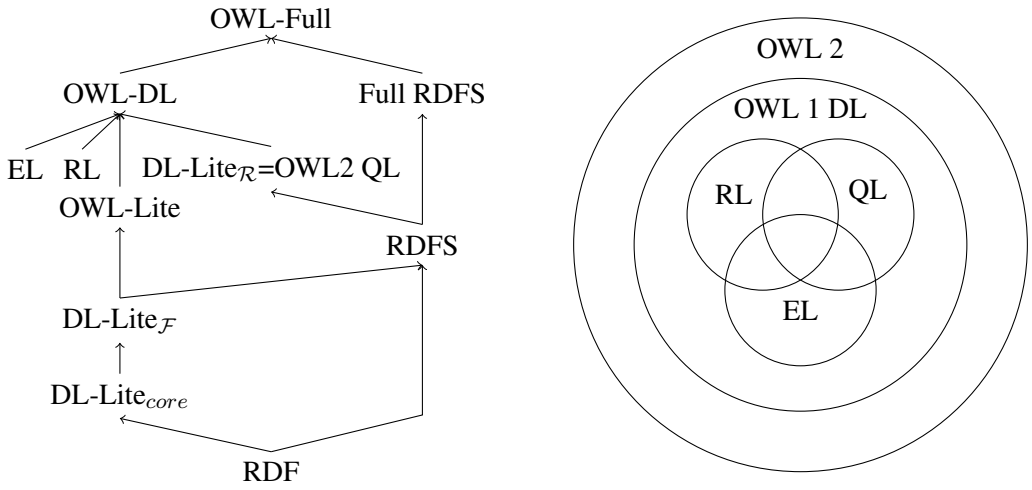


Figure 3.5: A hierarchy of OWL languages.

Inference engines have been implemented for significant subsets of OWL DL (in particular Pellet). Moreover there are still works for finding sweeter spots in the OWL language family. One of these is DL-Lite that is presented below.

3.2.5 DL-Lite a sub family on its own

DL-Lite [CALVANESE, DE GIACOMO, LEMBO, et al. 2007] is a family of description logics that has been designed for supporting efficient (polynomial) conjunctive query answering. It is based on the reduction of the expressivity of the OWL language. This is achieved by choosing sublanguages whose query evaluation can be reduced to simple relational database queries.

Definition 22 (DL-Lite syntax)**.** *DL-Lite terms are based on the following grammar:*

$$C \rightarrow B|\neg B$$
$$B \rightarrow A|\exists R$$
$$R \rightarrow P|P^{-1}$$

in which A and P are IRIs identifying concepts and properties respectively. The DL-Lite$_{core}$ language contains assertions of the form $B \sqsubseteq C$, in addition, DL-Lite$_\mathcal{R}$ authorises axioms like $R \sqsubseteq R$ and $R \sqsubseteq \neg R$, and DL-Lite$_\mathcal{F}$ added functionality axioms.

This corresponds to the use of the following OWL constructs: `rdfs:subClassOf`, `owl:minCardinality` (with value 1), `owl:Thing`, `owl:Nothing`, `owl:complementOf`, `owl:inverseOf`, to which is added the complement of a property. This simple language allows to express assertions like $B \sqsubseteq C \sqcap C'$ which is equivalent to $B \sqsubseteq C$ and $B \sqsubseteq C'$.

In fact, DL-Lite$_\mathcal{F}$ is a strict subset of OWL-Lite or RDFS (it cannot express `rdfs:range` or `owl:allValuesFrom`) and apparently DL-Lite$_\mathcal{R}$ is a strict superset of (the description logic part of) RDFS[1].

DL-Lite languages comply with the semantics of OWL, but for one thing: they obey the unique name assumption, hence it is not possible that two IRIs identify the same individual.

[1]The same authors went to introduce DL-Lite$_{\mathcal{FR}}$ as the union of DL-Lite$_\mathcal{F}$ and DL-Lite$_\mathcal{R}$ as well as DL-Lite$_\mathcal{A}$ as a subset of DL-Lite$_{\mathcal{FR}}$ with further restrictions such as the separation between roles and features [POGGI et al. 2008].

Example 12. *The example given in Marie-Christine Rousset's slides [*ABITEBOUL *et al. 2011]:*

$$
\begin{aligned}
Professor &\sqsubseteq \exists teachesTo \\
Student &\sqsubseteq \exists hasTutor \\
\exists teachesTo^{-1} &\sqsubseteq Student \\
\exists hasTutor^{-1} &\sqsubseteq Professor \\
Professor &\sqsubseteq \neg Student \\
hasTutor^{-1} &\sqsubseteq teachesTo && DL-Lite_{\mathcal{R}} \\
(func\ hasTutor) & && DL-Lite_{\mathcal{F}}
\end{aligned}
$$

can be expressed in OWL by:

```
<owl:Class rdf:about="#Professor">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#teachesTo" />
      <owl:minCardinality>1</owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <owl:disjointWith rdf:resource="#Student" />
</owl:Class>

<owl:Class rdf:about="#Student">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasTutor" />
      <owl:someValuesFrom rdf:resource="&owl;Thing" />
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

<owl:Class>
  <owl:equivalentClass>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty>
          <owl:inverseOf rdf:resource="#teachesTo" />
        </owl:ObjectProperty>
      </owl:onProperty>
      <owl:someValuesFrom rdf:resource="&owl;Thing">
    </owl:Restriction>
  </owl:equivalentClass>
  <rdfs:subClassOf rdf:resource="#Student" />
</owl:Class>

<owl:Class>
  <owl:equivalentClass>
    <owl:Restriction>
      <owl:onProperty>
        <owl:Property>
          <owl:inverseOf rdf:resource="#hasTutor" />
        </owl:Property>
      </owl:onProperty>
```

Problem	Axiom entailment	Conjunctive query
OWL Full	undecidable	undecidable
OWL 2	2NEXPTIME-complete	open
OWL 1 DL	NEXPTIME-complete	open (decidable)
OWL 1 Lite	EXPTIME-complete	
ρDF	NP-complete, PTIME (ground)	
OWL 2 EL	PTIME-complete	PSPACE-complete
OWL 2 RL	PTIME-complete	NP-complete
OWL 2 QL	NLOGSPACE-complete	NP-complete (UCQ), AC^0 (CQ)
DL-Lite$_{\mathcal{F}}$	PTIME	NP-complete (UCQ)
DL-Lite$_{core}$	PTIME	

Table 3.4: Worst-case combined complexity results for problems in OWL.

```
      <owl:someValuesFrom rdf:resource="&owl;Thing">
    </owl:Restriction>
  </owl:equivalentClass>
  <rdfs:subClassOf rdf:resource="#Professor" />
</owl:Class>

<owl:Property rdf:about="#hasTutor">
  <rdfs:subPropertyOf>
    <owl:ObjectProperty>
      <owl:inverseOf rdf:resource="#teachesTo" />
    </owl:ObjectProperty>
  </rdfs:subPropertyOf>
</owl:Propert>

<owl:FunctionalProperty rdf:about="#hasTutor" />
```

The definition of `hasTutor` *is correct because if* $r \sqsubseteq r'$*, then* $r^{-1} \sqsubseteq r'^{-1}$ *(proove it and see how it affect our definition).*

You can as well try to express this as a set of triples and check that the semantics for this piece of description logic is the same as the semantics for OWL.

3.2.6 Computational results

Table 3.4 summarises the results from [MOTIK, CUENCA GRAU, et al. 2009] and [CALVANESE, DE GIACOMO, LEMBO, et al. 2007] about the complexity of the axiom entailment problem (consistency, class satisfiability, class subsumption, instance checking) and conjunctive query answering.

Because we retained combined complexity, the choice of a particular logic may also depend on the dominant factor in the problem to be dealt with (the size of the data, the size of the query or that or the ontology). Of course, this choice should primarily depend on what is to be expressed.

3.3 Conclusion

Ontology languages allows for defining the vocabulary used in RDF graphs and for constraining the use of this vocabulary. Thanks to such constraints, it is possible to reason over RDF graphs and deduce entailed

information. Achieving completeness in reasoning, i.e., finding all and only what is entailed is not necessarily an easy task. Hence, different ontology languages of various expressiveness have been defined. It is necessary to choose the most adequate language for expressing an ontology.

Although, RDF Schema allows for defining simple ontologies, it does not allow for powerful constraints such as defining classes by disjunction or complement or by constraining their properties on type or cardinality. The OWL language introduces further vocabulary (and structure constraints) for defining more expressive ontologies. This led to extend further the semantic structure of these languages.

Entailment may be used when querying RDF graphs, i.e., for querying not just the graph but what it entails, and when connecting ontologies and RDF graphs together. This will be considered in the remaining part of this document.

3.4 Exercises

Exercise 8 (OWL ontologies)**.**

1. *Describe in OWL (RDF or XML/RDF) the ontology containing the following assertions:*
   - *All authors are persons;*
   - *A book (`m:Livre`) has exactly one year of publication (`m:annee`);*
   - *A novel (`m:Roman`) is a book (`m:Livre`) and a book is a work (`m:Oeuvre`);*
   - *The title (`dc:title`) of a work is a character string (`xsd:string`);*
   - *The relation "a écrit" (`m:aecrit`) relates an author to a work.*
2. *If one associates the graph (a) of Figure 2.6 of Exercise 1 (p26) and the ontology resulting from the previous question, is it possible to deduce the type (`rdf:type`) of `?x`?*

   Can you semantically justify how? What else can be deduced?
3. *Does the use of this ontology for defining the graphs of Figure 2.6 (p26) would change something to the answer to Question 4 of Exercise 1? What?*

Exercise 9 (DL-Lite ontologies)**.** *Consider the ontology O made of the following DL-Lite assertions:*

$$\begin{aligned} \texttt{worksWith} &\sqsubseteq \texttt{foaf:knows} \\ \texttt{playsTennisWith} &\sqsubseteq \texttt{playsSportWith} \\ \texttt{playsSportWith} &\sqsubseteq \texttt{foaf:knows} \\ \texttt{marriedWith} &\sqsubseteq \texttt{foaf:knows} \\ \texttt{Person} &\sqsubseteq \exists \texttt{foaf:mbox} \end{aligned}$$

1. *In which dialect (sublanguage) of DL-Lite is this ontology expressed (DL-Lite$_{core}$, DL-Lite$_{\mathcal{F}}$, DL-Lite$_{\mathcal{R}}$)?*
2. *Rewrite O in OWL or RDFS.*
3. *In which dialect (sublanguage) of RDFS or OWL is your ontology expressed?*
4. *Given the graph of Figure 2.7(b) of Exercise 2 (p27) to which the axioms of O are added. Compute its closure. What is the difference with the partial closure?*
5. *Given the graphs of Figure 2.7 (p27) to which the axioms of O are added, does this change something to the answers given to Question 3 of Exercise 2 if we consider RDFS-entailment? (explain why)*

Exercise 10 (OWL ontologies)**.**

1. *Provide an ontology satisfied by both graphs of Figure 2.7 of Exercise 2 (p27).*

We would like to express that an email address cannot correspond to more than one person by the property `foaf:mbox`*.*

2. *How to express this in OWL: with a cardinality constraint, a functional property, an inverse functional property or a symmetric property?*
3. *Is it possible to express this constraint in DL-Lite? If yes, how?*
4. *Consider the graphs of Figure 2.7 (p27) to which this constraint is added, does this change something to the answers given to Question 3 of Exercise 2? (explain why)*

Exercise 11 (RDFS ontologies)**.** *Consider the RDFS ontology o containing, in addition to those of the graph G of Exercise 3, the following statements:*

$$\langle \texttt{o:Novel}, \texttt{rdfs:subClassOf}, \texttt{o:Literature}\rangle$$
$$\langle \texttt{o:Poem}, \texttt{rdfs:subClassOf}, \texttt{o:Literature}\rangle$$
$$\langle \texttt{o:translated}, \texttt{rdfs:range}, \texttt{o:Literature}\rangle$$
$$\langle \texttt{o:wrote}, \texttt{rdfs:domain}, \texttt{o:Writer}\rangle$$

1. *Does this allow to conclude that* `d:Poe`*,* `d:Baudelaire` *or* `d:Mallarmé` *is a* `o:Writer`*? Explain why.*
2. *Can you express in OWL the statement that "anyone who write Literature is a Writer"?*

Exercise 12 (OWL 2 qualified cardinality restrictions)**.** *OWL 2 introduced qualified cardinality restrictions (*`owl:qualifiedCardinality`*,* `owl:maxQualifiedCardinality`*, and* `owl:minQualifiedCardinality`*, whose interpretation is obtained by extending the E_C function of Definition 19:*

$$\begin{aligned} & E_C(\texttt{restriction}(p, \texttt{minQualifiedCardinality}(n, C))) \\ = \ & \{x \in O; |\{\langle x, y\rangle \in E_R(p); y \in E_C(C)\}| \geq n\} \\ & E_C(\texttt{restriction}(p, \texttt{maxQualifiedCardinality}(n, C))) \\ = \ & \{x \in O; |\{\langle x, y\rangle \in E_R(p); y \in E_C(C)\}| \leq n\} \\ & E_C(\texttt{restriction}(p, \texttt{qualifiedCardinality}(n, C))) \\ = \ & \{x \in O; |\{\langle x, y\rangle \in E_R(p); y \in E_C(C)\}| = n\} \end{aligned}$$

Consider the following expressions (in OWL 2):

```
ex:SmallTeam rdfs:subClassOf _:a .
_:a rdf:type owl:Restriction .
_:a owl:onProperty ex:member .
_:a owl:maxCardinality 5 .
ex:ModernTeam2 rdfs:subClassOf ex:SmallTeam .
ex:ModernTeam2 rdfs:subClassOf _:b .
_:b rdf:type owl:Restriction .
_:b owl:onProperty ex:member .
_:b owl:minQualifiedCardinality 4 .
_:b owl:onClass ex:Woman .
```

1. *Draw the graph corresponding to this set of triples.*
2. *Express it in OWL/XML.*
3. *Explain the meaning of this graph (paraphrase it in English)*
4. *What would happen if we exchange the 5 and the 4?*

Exercise 13 (From OWL 2 to OWL 1 and back)**.**

1. *How is it possible to rewrite* `qualifiedCardinality` *in function of the minimal and maximal qualified cardinality restrictions? Explain it with the semantics.*
2. *Is it possible to express* `minCardinality`, `maxCardinality`, `cardinality`, `someValuesFrom` *with these new qualified cardinality restrictions? Explain how.*

Consider, in addition to the previous RDF graphs, the following statements (expressed in OWL 1):

```
ex:womanmember owl:subPropertyOf ex:member .
ex:womanmember rdfs:range ex:Woman .
ex:ModernTeam1 rdfs:subClassOf ex:SmallTeam .
ex:ModernTeam1 rdfs:subClassOf _:b .
_:b rdf:type owl:Restriction .
_:b owl:onProperty ex:womanmember .
_:b owl:minCardinality 4 .
```

3. *Does* `ex:ModernTeam1` *subsume* `ex:ModernTeam2` *or the other way around? Justify.*
4. *Does this suggest that it is also possible to express qualified cardinality constraints in OWL 1? Explain.*
5. *Does qualified cardinality restrictions provide additional expressivity to OWL 1?*

Exercise 14 (FRBR in RDFS)**.** *FRBR is now a well established vocabulary in libraries. FRBR distinguishes between:*

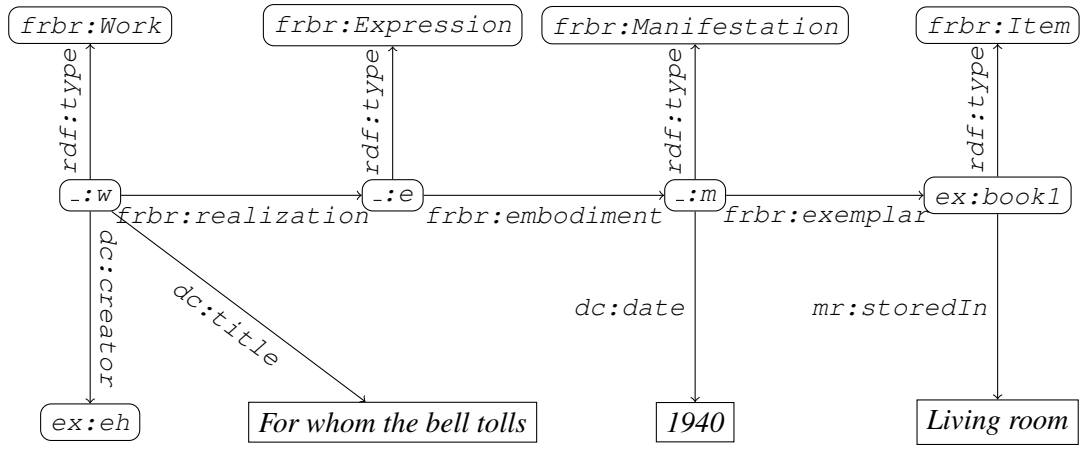
- *a* work *which is an abstract idea of what a work is;*
- *an* expression *which is a realization of this work in a particular form (poetry, music, painting);*
- *a* manifestation *which is a distinct embodiment of the expression (an edition of a text, a release of a movie; the reproduction of a painting);*
- *an* item *which is an often physical exemplar of a manifestation (an exemplar of a book; a copy of an MP3 file).*

Consider a fragment S of its RDF Schema manifestation:

```
frbr:Work rdf:type rdfs:Class .
frbr:Expression rdf:type rdfs:Class .
frbr:Manifestation rdf:type rdfs:Class .
frbr:Item rdf:type rdfs:Class .
frbr:Performance rdfs:subClassOf frbr:Expression .

frbr:adaptation rdfs:domain frbr:Work .
frbr:adaptation rdfs:range frbr:Work .
frbr:realization rdfs:domain frbr:Work .
frbr:realization rdfs:range frbr:Expression .
frbr:translation rdfs:domain frbr:Expression .
```

```
frbr:translation rdfs:range frbr:Expression .
frbr:embodiment rdfs:domain frbr:Expression .
frbr:embodiment rdfs:range frbr:Manifestation .
frbr:exemplar rdfs:domain frbr:Manifestation .
frbr:exemplar rdfs:range frbr:Item .
```

For instance, the first 5 statements of the graph G of Exercise 4 in FRBR would correspond to the following graph:



1. *How would you extend the above S with the vocabulary identified by the prefix* `mr` *in graph G of Exercise 4? I.e., give triples using the RDFS vocabulary (*`rdfs:subClassOf`*,* `rdfs:subPropertyOf`*,* `rdfs:domain`*,* `rdfs:range`*) to extend FRBR with the corresponding entities.*

Exercise 15 (RDFS Entailment)**.** *Given* P'=$\{$ *?x* `rdf:type` `a:Hospital` . *?x* `a:hasService` *?y* . *?y* `rdf:type` `a:Maternity` . *?x* `a:hasService` *?z* . *?z* `rdf:type` `a:Pediatry` .$\}$ *and G the graph of Figure 2.8 (p. 29),*

1. *Does* $G \models_{RDF} P'$*? (explain how and/or why)*
2. *Which procedure may be used to derive that* $G \models_{RDFS} P'$*?*

Exercise 16 (RDFS and OWL interpretation)**.**

1. *One convenient way to interpret together two heterogeneous sources is to interpret them through a common ontology. Consider the ontology O made of the following statements:*

   ```
   o:parent rdfs:domain foaf:Person .
   o:parent rdfs:range foaf:Person .
   o1:mother rdfs:subPropertyOf o:parent .
   o1:father rdfs:subPropertyOf o:parent .
   o1:mother rdfs:domain o2:Female .
   o1:father rdfs:domain o2:Male .
   ```

   Considering graphs G_1*,* G_2*, and* Q_1 *of Exercise 6 (p.28), does* $O \cup G_1 \cup G_2 \models_{RDF} Q_1$*?*

2. *Does* $O \cup G_1 \cup G_2 \models_{RDFS} Q_1$*? (explain your answer) Give all mappings (variable/blank assignments) which support this entailment. What additional facts does* $O \cup G_1 \cup G_2$ *RDFS-entail? (provide an example).*

3. *Can you express in OWL the class* `o:ParentOfNumerousChildren`*, as the class of those parents with more than three children, using the concepts and properties of ontology O? Give the interpretation of this (compound) class.*

Exercise 17 (RDFS and OWL entailment)**.** *Consider the ontology O made of the following statements:*

```
o:accomodation rdfs:range o:Accomodation .
o:Local rdfs:subClassOf o:Accomodation .
o:Pousada rdfs:subClassOf o:Local .
o:Ryokan rdfs:subClassOf o:Local .
o:GrandHotel rdfs:subClassOf Accomodation .

o:activity rdfs:range o:Activity .
o:Sport rdfs:subClassOf o:Activity .
o:Swimming rdfs:subClassOf o:Sport .
o:SwordFighting rdfs:subClassOf o:Sport .
o:Visit rdfs:subClassOf o:Activity .
o:Cruising rdfs:subClassOf o:Visit .
```

and the graph G of Exercise 7

1. *Does $G \models_{RDFS}$ `o:Package rdf:type rdfs:Class`?*
   Does $O \models_{RDFS}$ `o:Package rdf:type rdfs:Class`?
2. *Does $O \cup G \models_{RDF} K$? $O \cup G \models_{RDFS} K$? Explain why.*
3. *Given the OWL axiom (making the OWL ontology O'):*

$$\texttt{o:TonicPackage} \equiv \texttt{o:Package} \sqcap \exists \texttt{o:accomodation}.(\texttt{o:Local} \sqcap \geq_1 \texttt{o:swimmingPool}) \sqcap \exists \texttt{o:activity}.\texttt{o:Sport}$$

   Give the OWL interpretation of TonicPackage ($E_C($`o:TonicPackage`$)$).
4. *Does $O \cup O' \cup G \models_{OWL}$ `_:b1 rdf:type o:TonicPackage`? Tell why.*

Rules

Concerning rules, the situation is less clear than for other semantic web languages. The W3C has recommended a '*rule interchange format*' (RIF [BOLEY et al. 2013]) whose goal is to offer both logic programming rules and '*business rules*', a kind of rule triggering an action when a condition is satisfied. Two dialects, sharing part of their syntax (RIF-Core), have been defined: BLD for '*Basic Logic Dialect*' and PRD for '*Production Rule Dialect*'. RIF has a syntax different from those provided by other languages. In particular, the RDF syntax of RuleML and SWRL [HORROCKS, PATEL-SCHNEIDER, BOLEY, et al. 2004] has not been retained for RIF.

From a theoretical standpoint, all work developed in logic programming applies naturally to the semantic web and can be transposed in the RIF-BLD dialect. Moreover, some connection between rules and ontologies have been specified [DE BRUIJN and WELTY 2013].

Combining description logics and Datalog has been studied [LEVY and ROUSSET 1998]. In the semantic web context, work has started from less expressive fragments: DLP ('*description logic programs*' [GROSOF et al. 2003]) is a minimal language combining Horn clauses and description logics. Hence, it does not allow for expressing neither all OWL Lite, nor all logic programming. However, it could be adopted to reason at large scale. DLP has been the inspiration for OWL 2 RL.

A closer integration of description logics and of '*answer set programming*', which reintroduces aspects such as negation-as-failure and the closed world assumption, has also been considered [EITER et al. 2008; MOTIK and ROSATI 2010]. Other work have attempted the definition of description logics on a logic programming language basis [HUSTADT et al. 2007; HUSTADT et al. 2008].

From the conceptual graph side, logical rules have been studied as an extension of simple conceptual graphs [BAGET et al. 2011] which can be directly applied to RDF given the proximity of these formalisms.

Part II

Queries

Chapter 4

Querying the semantic web

4.1 Motivation

We know how to express information on the web. It is now necessary to take advantage of it.

Nowadays, more resources are annotated via RDF due to its simple data model, formal semantics, and sound and complete inference mechanisms. RDF itself can be used as a query language for an RDF knowledge base using RDF entailment test. Nonetheless, the use of consequence is still limited for answering queries. In particular, answering those that contain complex relations requires complex constructs. It is impossible, for example, to answer the query "find the names and addresses, if they exist, of persons who either work on query languages or ontology matching" using a simple entailment test.

Therefore the need for added expressivity in queries has led to define several query languages on top of graph patterns that are basically RDF and more precisely GRDF graphs. The focus of the next chapters is then to give an overview of some languages that have been designed or can be used for querying RDF graphs, and discusses the main differences between them in terms of expressiveness and limitations.

4.2 What is a query language?

Basically, the goal of a query language is to express queries that are evaluated against a representation. Query evaluation covers various aspects:

- Extracting information from a structure;
- Deducing information from a representation;
- Aggregating query answers;
- Constructing new structure.

4.2.1 Data extraction languages

The basic data extraction paradigm is the web. You know the URL, you got the HTML page. This us very basic and not very well structured: the operations for combining HTML pages are not very natural.

In the XML world, XPath is the appropriate language for extracting information. XML documents and answers are node sets to which queries can be applied and hence composed. XPath allows to navigate in the structure and offers powerful selection operations.

4.2.2 Data manipulation language

Data manipulation languages can do more than extraction languages. Following SQL, XQuery and XSLT typically provide XML as output. Not only they evaluate XPath queries for extracting information in XML

documents, but they take advantage of the answers for constructing new XML documents. These documents can be fed again in XQuery or XSLT engines.

4.2.3 Query language as an algebra: SQL

A typical aspect of SQL is to be itself an algebra, i.e., queries are expression of an algebra that can be manipulated according to the rules of the algebra. This is used for distributing or rewriting queries for optimisation (with regard to a schema for instance).

4.2.4 Querying as testing consequences

So far, we did not consider deducting consequence from the representation: a query language is only here to extract and transform information.

Since we provided a semantics for semantic web languages, queries should be evaluated with regard to this semantics. So instead of merely *extracting* information from a structure, the basic operation of a query language should be to know if information is consequence of the data. Since we provided different logics for expressing knowledge on the web, it is natural to be able to parameter queries with regard to these different logics (and the kind of entailment they offer). Hence, querying semantic web data can be expressed by:

$$\text{Data + Ontology} \models_L \phi$$

such that L is a particular representation language (Simple RDF, RDF, GRDF, RDF Schema, OWL Lite, OWL DL, OWL Full or others). In our case, the formula ϕ will be based on RDF graphs.

Chapter 5

Querying RDF with SPARQL

There has been early proposals for specific RDF query languages, such as RDQL [SEABORNE 2004], RQL [KARVOUNARAKIS et al. 2002] or SeRQL [BROEKSTRA 2003]. In 2004, the W3C launched the Data Access Working Group for designing an RDF query language, called SPARQL, from these early attempts [PRUD'HOMMEAUX and SEABORNE 2008]. SPARQL is inspired by the SQL relational data base query language.

We define in the following subsections the syntax and the semantics of SPARQL. For a complete description of SPARQL, the reader is referred to the SPARQL specification [PRUD'HOMMEAUX and SEABORNE 2008] or to [PÉREZ et al. 2009; POLLERES 2007] for its formal semantics. Unless stated otherwise, we concentrates on SPARQL 1.0, but some features considered in the presented languages are now integrated in SPARQL 1.1 [HARRIS and SEABORNE 2013; GLIMM and OGBUJI 2013].

5.1 Syntax

In the following, we treats blank nodes in RDF simply as constants (as if they were IRIs) as done in the official specification of SPARQL without considering their existential semantics. However, if the existential semantics of blank nodes is considered when querying RDF, the results of this paper indirectly apply by using the graph homomorphism technique [BAGET 2005].

5.1.1 SPARQL graph patterns

The heart of SPARQL queries is graph patterns. Informally, a *graph pattern* can be one of the following (see [PRUD'HOMMEAUX and SEABORNE 2008] for more details):

- a **triple pattern**: a triple pattern corresponds in RDF to a GRDF triple;
- a **basic graph pattern**: a set of triple patterns (or a GRDF graph) is called a basic graph patterns;
- a **union of graph patterns**: we use the keyword `UNION` in SPARQL to represent alternatives;
- an **optional graph pattern**: SPARQL allows optional results to be returned determined by the keyword `OPTIONAL`;
- a **constraint**: constraints in SPARQL are boolean-valued expressions that limit the number of answers to be returned. They can be defined using the keyword `FILTER`. As atomic `FILTER` expressions, SPARQL allows unary predicates like `BOUND`; binary (in)equality predicates ($=$ and $!=$); comparison operators like $<$; data type conversion and string functions which will be omitted here. Complex `FILTER` expressions can be built using !, || and &&;
- a **group graph pattern**: is a graph pattern grouped inside { and }, and determines the scope of SPARQL constructs like `FILTER` and variable nodes;

Definition 23 (SPARQL graph pattern)**.** *A* SPARQL graph pattern *is defined inductively in the following way:*

- *every GRDF graph is a SPARQL graph pattern;*
- *if P, P' are SPARQL graph patterns and K is a SPARQL constraint, then (P AND P'), (P UNION P'), (P OPT P'), and (P FILTER K) are SPARQL graph patterns.*

A SPARQL constraint K is a boolean expression involving terms from $(\mathcal{V} \cup \mathcal{B})$, e.g., a numeric test. We do not specify these expressions further (see [MUÑOZ et al. 2009] for a more complete treatment).

Example 13. *The following graph pattern:*

```
{ ?person foaf:knows "Faisal" . }
```

is a basic graph pattern that can be used in a query for finding persons who know Faisal.

```
{
   { ?person ex:liveIn ex:France . }
      UNION
   { ?person ex:hasNationality ex:French . }
}
```

is a union of two basic graph patterns that searches the persons who either live in France or have a French nationality.

The following graph pattern

```
{    ?person foaf:knows "Faisal" .
       OPTIONAL
   { ?person foaf:mbox ?mbox . }
}
```

contains an optional basic graph pattern searching the mail boxes, if they exist, of persons who know Faisal.

```
{
    ?person ex:liveIn ex:France .
    ?person ex:hasAge ?age .
    FILTER ( ?age < 40 ) .
}
```

the constraint in this graph pattern limits the answers to the persons who live in France whose ages are less than 40*.*

```
{ { ?person foaf:knows "Faisal" . }
  {
    ?person ex:liveIn ex:France .
    ?person ex:hasAge ?age .
    FILTER ( ?age < 40 ) .
  }
}
```

is a graph pattern of two group graph patterns. The scope of the constraint in this graph pattern is the second group graph pattern. So, it is applied only to the persons who live in France.

5.1.2 SPARQL query

SPARQL provides several query forms.

Definition 24 (SPARQL query)**.** *Given a SPARQL graph pattern P, a tuple $\vec{B}$ of variable in P, a IRI u and a basic graph pattern Q,*

$$\texttt{ASK FROM } u \texttt{ WHERE } P$$
$$\texttt{SELECT } \vec{B} \texttt{ FROM } u \texttt{ WHERE } P$$
$$\texttt{CONSTRUCT } Q \texttt{ FROM } u \texttt{ WHERE } P$$

are SPARQL queries.

Intuitively, an answer to a SELECT query is an assignment of the variables in $\vec{B}$ by the terms of the RDF graph G such that, under these assignments, P is entailed by the graph identified by u. A CONSTRUCT query is used for building an RDF graph from the set of answers. ASK returns `TRUE` if there is a answer to a given query and `FALSE` otherwise. In addition, DESCRIBE is used for describing a resource RDF graph.

The following example queries give an insight of these query forms.

Example 14 (Query)**.** *The following query searches, in the regulatory network of Figure 2.1, a gene $?x$ which inhibits a product that regulates a product that $?x$ promotes, and returns these three entities:*

```
SELECT ?x, ?y, ?z
WHERE
  ?x rn:inhibits ?y
  ?x rn:promotes ?z
  ?y rn:regulates ?z
  ?x rdf:type rn:gene.
```

The following ASK query:

```
ASK
WHERE { ?person foaf:name "Faisal" .
        ?person ex:hasChild ?child  .
      }
```

returns `TRUE` if a person named Faisal has at least one child, `FALSE` otherwise.

The following CONSTRUCT query:

```
CONSTRUCT { ?son1 ex:brother ?son2 .}
WHERE {
            ?son1 ex:sonOf ?person .
            ?son2 ex:sonOf ?person .
            FILTER ( ?son1 != ?son2 ) .
      }
```

constructs the RDF graph (containing the brotherhood relation) by substituting for each located answer the values of the variables $?son1$ and $?son2$.

The following query:

```
DESCRIBE <example.org/person1>
```

returns a description of the resource identified by the given IRI, i.e., returns the set of triples involving this IRI.

SPARQL uses post-filtering clauses which allow, for example, to order (ORDER BY clause with modifiers ASC/DSC), to group (GROUP BY) or to limit (LIMIT and/or OFFSET clauses) the answers of a query. The reader is referred to the SPARQL specification [PRUD'HOMMEAUX and SEABORNE 2008] for more details or to [PÉREZ et al. 2009] for formal semantics of SPARQL queries.

Example 15 (Post-filtered SPARQL query)**.** *The following SPARQL query:*

```
SELECT  ?name
WHERE  {
        ?person ex:liveIn ex:France .
        ?person foaf:name ?name .
       }
ORDER BY ?name ASC
LIMIT 10
OFFSET 5
```

returns the names of persons who live in France limited to maximum 10 *persons, ordered by their names, and starting from the* 5*th answer.*

SPARQL 1.1 [HARRIS and SEABORNE 2013] also introduced the opportunity to combine queries and sources. In particular, queries can integrate subqueries, evaluate a query part againts a specific named graph (with the FROM NAMED and GRAPH constructs), and evaluate federated queries, i.e., queries against several data sources whose results are combined (with the SERVICE construct).

Since the graph patterns in the SPARQL query language are shared by all SPARQL query forms, we illustrate them using the SELECT ... FROM ... WHERE ... queries.

5.2 SPARQL Semantics

In the following, we characterize query answering with SPARQL as done in [PÉREZ et al. 2009]. The approach relies upon the correspondence between GRDF entailment and maps from RDF graph of the query graph patterns to the RDF knowledge base (see Definition 8, p.24).

Definition 25 (Application of a map to a basic graph pattern)**.** *The application $\sigma(P)$ of a map σ to a basic graph pattern P, is defined by:*

- $\sigma(P) = \{\sigma(t); t \in P\}$ *if P is a GRDF graph;*
- $\sigma(\langle s, p, o\rangle) = \langle \sigma'(s), \sigma'(p), \sigma'(o)\rangle$ *if P is a triple;*
- $\sigma'(x) = \sigma(x)$ *if $x \in dom(\sigma)$;*
- $\sigma'(x) = x$ *otherwise.*

SPARQL query constructs are defined through algebraic operations on maps: assignments from a set of variables to terms that preserve names.

Operations on maps If σ is a map, then the domain of σ, denoted by $dom(\sigma)$, is the subset of $\mathcal{T}$ on which σ is defined. The restriction of σ to a set of terms X is defined by $\sigma|_X = \{\langle x, y\rangle \in \sigma |\ x \in X\}$ and the completion of σ to a set of terms X is defined by $\sigma|^X = \sigma \cup \{\langle x, \texttt{null}\rangle |\ x \in X \text{ and } x \notin dom(\sigma)\}$[1].

If P is a graph pattern, then $\mathcal{B}(P)$ is the set of variables occurring in P and $\sigma(P)$ is the graph pattern obtained by the substitution of $\sigma(b)$ to each variable $b \in \mathcal{B}(P)$. Two maps σ_1 and σ_2 are *compatible* when $\forall x \in dom(\sigma_1) \cap dom(\sigma_2)$, $\sigma_1(x) = \sigma_2(x)$. Otherwise, they are said to be incompatible and this is denoted by $\sigma_1 \perp \sigma_2$. If σ_1 and σ_2 are two compatible maps, then we denote by $\sigma = \sigma_1 \oplus \sigma_2 : T_1 \cup T_2 \to \mathcal{T}$ the map defined by $\forall x \in T_1, \sigma(x) = \sigma_1(x)$ and $\forall x \in T_2, \sigma(x) = \sigma_2(x)$.

In the following, we use an alternate characterization of SPARQL query answering that relies upon the correspondence between GRDF entailment and maps from the query graph patterns to the RDF graph [PÉREZ et al. 2009]. The answers to a basic graph pattern query are those maps which warrant the entailment of the graph pattern by the queried graph. In the case of SPARQL, this entailment relation is GRDF entailment.

[1] The `null` symbol is used for denoting the NULL values introduced by the OPTIONAL clause.

Definition 26 (Graph pattern entailment)**.** *Let $\models$ be an entailment relation on basic graph patterns, P, P' be SPARQL graph patterns, K be a SPARQL constraint, and G be an RDF graph,* graph pattern entailment *by an RDF graph modulo a map σ is defined inductively by:*

$$
\begin{aligned}
G \models \sigma(P \texttt{ AND } P') &\textit{ iff } G \models \sigma(P) \textit{ and } G \models \sigma(P') \\
G \models \sigma(P \texttt{ UNION } P') &\textit{ iff } G \models \sigma(P) \textit{ or } G \models \sigma(P') \\
G \models \sigma(P \texttt{ OPT } P') &\textit{ iff } G \models \sigma(P) \textit{ and } [G \models \sigma(P') \textit{ or } \forall \sigma'; G \models \sigma'(P'), \sigma \perp \sigma'] \\
G \models \sigma(P \texttt{ FILTER } K) &\textit{ iff } G \models \sigma(P) \textit{ and } \sigma(K) = \top
\end{aligned}
$$

The conditions K are interpreted as boolean functions from the terms they involve. Hence, $\sigma(K) = \top$ means that this function is evaluated to true once the variables in K are substituted by σ. If not all variables of K are bound, then $\sigma(K) \neq \top$. One particular operator that can be used in SPARQL filter conditions is "$bound(?x)$". This operator returns true if the variable $?x$ is bound and in this case $\sigma(K)$ is not true whenever a variable is not bound.

The semantics of SPARQL `FILTER` expressions is defined as follows: given a map σ and a SPARQL constraint K, we say that σ satisfies K (denoted by $\sigma(K) = \top$), if:

- $K = \texttt{BOUND}(x)$ with $x \in dom(\sigma)$;
- $K = (x = c)$ with $x \in dom(\sigma)$ and $\sigma(x) = c$;
- $K = (x = y)$ with $x, y \in dom(\sigma)$ and $\sigma(x) = \sigma(y)$;
- $K = (x != c)$ with $x \in dom(\sigma)$ and $\sigma(x) != c$;
- $K = (x != y)$ with $x, y \in dom(\sigma)$ and $\sigma(x) != \sigma(y)$;
- $K = (x < c)$ with $x \in dom(\sigma)$ and $\sigma(x) < c$;
- $K = (x < y)$ with $x, y \in dom(\sigma)$ and $\sigma(x) < \sigma(y)$;
- $K = !K_1$ with $\sigma(K_1) = \perp$ (σ does not satisfy K_1);
- $K = (K_1 || K_2)$ with $\sigma(K_1) = \top$ or $\sigma(K_2) = \top$;
- $K = (K_1 \&\& K_2)$ with $\sigma(K_1) = \top$ and $\sigma(K_2) = \top$;

The definition is given with respect to an entailment relation $\models$ which can be replaced by a concrete relation defined on the base case, i.e., basic graph patterns.

As usual for this kind of query language, an answer to a query is an assignment of distinguished variables (those variables in the `SELECT` part of the query). Such an assignment is a map from variables in the query to nodes of the graph. The defined answers may assign only one part of the variables, those sufficient to prove entailment. The answers are these assignments extended to all distinguished variables.

Definition 27 (Answer to a SELECT SPARQL query [ALKHATEEB, BAGET, et al. 2009])**.** *Let $\texttt{SELECT } \vec{B} \texttt{ FROM } u \texttt{ WHERE } P$ be a SPARQL query with P a SPARQL graph pattern and G be the (G)RDF graph identified by the IRI u, then the set of answers to this query is*

$$\mathcal{A}(\vec{B}, G, P) = \{\sigma|^{\vec{B}}_{\vec{B}} \mid G \models_{GRDF} \sigma(P)\}.$$

This definition is a semantic characterization of SPARQL answers.

The completion to null does not prevent that blank nodes remain in answers: null values only replace unmatched variables. [POLLERES 2007] defines a different semantics for the join operation when the maps contain null values. This semantics could be adopted instead without changing the remainder.

Example 16 (Query evaluation)**.** *The evaluation of the query of Example 14 against the RDF graph of Example 2 returns only one answer:*

$$\langle \texttt{dm:bcd}, \texttt{dm:tll}, \texttt{dm:Kr} \rangle$$

Definition 28 (Answer to a ASK/CONSTRUCT SPARQL query)**.** *Let* `ASK FROM` u `WHERE` P *be a SPARQL query with* P *a SPARQL graph pattern and* G *be the (G)RDF graph identified by the IRI* u*, then the (boolean) answer to this query is*

$$\mathcal{A}^{ASK}(G, P) = \exists\sigma;\ G \models_{GRDF} \sigma(P)$$

Let `CONSTRUCT` G' `FROM` u `WHERE` P *be a SPARQL query with* P *a SPARQL graph pattern,* G' *a (G)RDF graph, and* G *be the (G)RDF graph identified by the IRI* u*, then the answer to this query is*

$$\mathcal{A}^{CONST}(G, P) = \bigcup_{\sigma;\ G \models_{GRDF} \sigma(P)} \sigma(G')$$

Example 17 (Query evaluation)**.** *The following query:*

```
SELECT ?z ?y ?k
WHERE  {
        { ?x ex:hasDaughter ?y .  } UNION { ?x ex:hasSon ?y . }
AND     { ?x ex:hasSon ?z .  } UNION { ?x ex:hasDaughter ?z . }
AND     { ?w ex:hasChild ?z .  ?v ex:hasChild ?y . }
FILTER { ?w != ?v && ?v != ?x && ?x != ?w }
OPTIONAL { ?x foaf:lastname ?k . }
}
```

to be evaluated on:

```
ex:Peter foaf:lastname "MacCartney" .
ex:Peter ex:hasDaughter ex:Mary . ex:Shirley ex:hasChild ex:Mary .
ex:Peter ex:asSon ex:Paul . ex:Kate ex:hasChild ex:Paul .
ex:Peter ex:hasDaughter ex:Julia. ex:Kate ex:hasDaughter ex:Julia .
```

will return the value tuples:

```
ex:Mary ex:Paul "MacCartney"
ex:Paul ex:Mary "MacCartney"
```

if used with the query type:

```
CONSTRUCT { ?z ex:hasHalfSibling ?y . }
```

it will add the triples:

```
ex:Mary ex:hasHalfSibling ex:Paul .
ex:Paul ex:hasHalfSibling ex:Mary .
```

Does it seems to be the correct set of siblings? Is there a problem to fix?

In order to evaluate the complexity of query answering, we use the following problem, usually named query evaluation but better named ANSWER CHECKING:
Problem: $\mathcal{A}$-ANSWER CHECKING
Input: an RDF graph G, a SPARQL graph pattern P, a tuple of variables $\vec{B}$, and a map σ.
Question: Does $\sigma \in \mathcal{A}(\vec{B}, G, P)$?

This problem has usually the same complexity as checking if an answer exists. For SPARQL, the problem has been shown to be PSPACE-complete.

Proposition 7 (Complexity of $\mathcal{A}$-ANSWER CHECKING [PÉREZ et al. 2009])**.** $\mathcal{A}$-ANSWER CHECKING *is PSPACE-complete.*

The complexity of checking RDF-entailment and GDRF-entailment is NP-complete [GUTIERREZ et al. 2004]. This means that $\mathcal{A}$-ANSWER CHECKING when queries are reduced to basic graph patterns is NP-complete. In fact, the addition of AND, FILTER, and UNION does not increase complexity which remains NP-complete. The PSPACE complexity comes from the addition of the OPT construction [PÉREZ et al. 2009].

Hence, for every language in which entailment is NP-complete, used as a basic graph pattern language, the problem for this language will have the same complexity since Definition 26 shows the independence of subquery evaluation.

5.3 Algebraic manipulation

Answering SPARQL queries may be obtained by directly manipulating graphs and maps. The original semantics of SPARQL was given in this way. For defining these manipulations, we need some further definition coming from relational database theory. The *join* and *difference* of two sets of maps Ω_1 and Ω_2 are defined as follows [PÉREZ et al. 2009]:

- *(join)* $\Omega_1 \bowtie \Omega_2 = \{\sigma_1 \oplus \sigma_2 \mid \sigma_1 \in \Omega_1, \sigma_2 \in \Omega_2 \text{ are compatible}\}$;
- *(difference)* $\Omega_1 \setminus \Omega_2 = \{\sigma_1 \in \Omega_1 \mid \forall \sigma_2 \in \Omega_2, \sigma_1 \text{ and } \sigma_2 \text{ are not compatible}\}$.

The answers to a basic graph pattern query are those maps which warrant the entailment of the graph pattern by the queried graph. In the case of SPARQL, this entailment relation is RDF-entailment. Answers to compound graph patterns are obtained through the operations on maps.

Definition 29 (Answers to compound graph patterns)**.** *Let $\models_{reg}$ be an entailment relation on basic graph patterns, P, P' be SPARQL graph patterns, K be a SPARQL constraint, and G be an RDF graph. The set $\mathcal{S}(P,G)$ of* answers *to P in G is the set of maps from $\mathcal{B}(P)$ to $\mathcal{T}(G)$ defined inductively in the following way:*

$$\begin{aligned}
\mathcal{S}_{reg}(P,G) &= \{\sigma|_{\mathcal{B}(P)} \mid G \models_{reg} \sigma(P)\} \quad \textit{if } P \textit{ is a basic graph pattern} \\
\mathcal{S}_{reg}((P \texttt{ AND } P'),G) &= \mathcal{S}_{reg}(P,G) \bowtie \mathcal{S}_{reg}(P',G) \\
\mathcal{S}_{reg}(P \texttt{ UNION } P',G) &= \mathcal{S}_{reg}(P,G) \cup \mathcal{S}_{reg}(P',G) \\
\mathcal{S}_{reg}(P \texttt{ OPT } P',G) &= (\mathcal{S}_{reg}(P,G) \bowtie \mathcal{S}_{reg}(P',G)) \cup (\mathcal{S}_{reg}(P,G) \setminus \mathcal{S}_{reg}(P',G)) \\
\mathcal{S}_{reg}(P \texttt{ FILTER } K,G) &= \{\sigma \in \mathcal{S}_{reg}(P,G) \mid \sigma(K) = \top\}
\end{aligned}$$

The following corollary is obtained thanks to the interpolation lemma and its application to homomorphisms.

Corollary 8. *Let* `SELECT` $\vec{B}$ `FROM` u `WHERE` P *be a SPARQL query with P a basic graph pattern, G be the RDF graph identified by the IRI u, and $\mathcal{S}_{GRDF}(P,G)$ be the set of answers to P in G using GRDF-entailment, then*

$$\mathcal{A}(\vec{B},G,P) = \{\sigma|^{\vec{B}}_{\vec{B}} \mid \sigma \in \mathcal{S}_{GRDF}(P,G)\}.$$

Proof. By the interpolation lemma, if there exists an homomorphism σ from P to G, then $G \models_{GRDF} P$, moreover, this homomorphism determines an instance of G, hence $G \models_{GRDF} \sigma(P)$. Hence, $\mathcal{S}(P,G) = \{\sigma|_{\mathcal{B}(P)} \mid G \models_{GRDF} \sigma(P)\}$, so $\{\sigma|^{\vec{B}}_{\vec{B}} \mid \sigma \in \mathcal{S}(P,G)\} = \{\sigma|^{\vec{B}}_{\vec{B}} \mid G \models_{GRDF} \sigma(P)\}$ which is exactly $\mathcal{A}(\vec{B},G,P)$. □

This corollary can be extended to any graph patterns by induction on the structure of graph patterns.

Proposition 9 (Answers to a SPARQL query [ALKHATEEB and EUZENAT 2013])**.** *Let* `SELECT` $\vec{B}$ `FROM` u `WHERE` P *be a SPARQL query, G be the RDF graph identified by the IRI u, and $\mathcal{S}_{GRDF}(P,G)$ be the set of answers to P in G using GRDF-entailment, then the* answers *$\mathcal{A}(\vec{B},G,P)$ to the query are the restriction and completion to $\vec{B}$ of answers to P in G, i.e.,*

$$\mathcal{A}(\vec{B},G,P) = \{\sigma|^{\vec{B}}_{\vec{B}} \mid \sigma \in \mathcal{S}_{GRDF}(P,G)\}.$$

Proof. The proof is obtained by induction: the base case is that of Corollary 8:

$$\sigma \in \mathcal{S}_{GRDF}(P,G) \Leftrightarrow G \models_{GRDF} \sigma(P) \quad \text{if } P \text{ is a basic graph pattern}$$

The induction step is based on the idea that if the base step is true of the subcomponents (P and P'), then:

$$
\begin{aligned}
\sigma \in \mathcal{S}_{reg}((P \texttt{ AND } P'), G) &\Leftrightarrow \sigma \in \mathcal{S}_{reg}(P, G) \bowtie \mathcal{S}_{reg}(P', G) \\
&\Leftrightarrow \sigma \in \mathcal{S}_{reg}(P, G) \text{ and } \sigma \in \mathcal{S}_{reg}(P', G) \\
&\Leftrightarrow G \models_{reg} \sigma(P) \text{ and } G \models_{reg} \sigma(P') \\
&\Leftrightarrow G \models_{reg} \sigma(P \texttt{ AND } P') \\
\sigma \in \mathcal{S}_{reg}(P \texttt{ UNION } P', G) &\Leftrightarrow \sigma \in \mathcal{S}_{reg}(P, G) \cup \mathcal{S}_{reg}(P', G) \\
&\Leftrightarrow \sigma \in \mathcal{S}_{reg}(P, G) \text{ or } \sigma \in \mathcal{S}_{reg}(P', G) \\
&\Leftrightarrow G \models_{reg} \sigma(P) \text{ or } G \models_{reg} \sigma(P') \\
&\Leftrightarrow G \models_{reg} \sigma(P \texttt{ UNION } P') \\
\sigma \in \mathcal{S}_{reg}(P \texttt{ OPT } P', G) &\Leftrightarrow \sigma \in (\mathcal{S}_{reg}(P, G) \bowtie \mathcal{S}_{reg}(P', G)) \cup (\mathcal{S}_{reg}(P, G) \setminus \mathcal{S}_{reg}(P', G)) \\
&\Leftrightarrow \sigma \in \mathcal{S}_{reg}(P, G) \bowtie \mathcal{S}_{reg}(P', G) \text{ or } \sigma \in \mathcal{S}_{reg}(P, G) \setminus \mathcal{S}_{reg}(P', G) \\
&\Leftrightarrow (\sigma \in \mathcal{S}_{reg}(P, G) \text{ and } \sigma \in \mathcal{S}_{reg}(P', G)) \\
&\qquad \text{or } (\sigma \in (\mathcal{S}_{reg}(P, G) \text{ and } \sigma \notin \mathcal{S}_{reg}(P', G)) \\
&\Leftrightarrow (G \models_{reg} \sigma(P) \text{ and } G \models_{reg} \sigma(P')) \text{ or } (G \models_{reg} \sigma(P) \text{ and } G \not\models_{reg} \sigma(P')) \\
&\Leftrightarrow (G \models_{reg} \sigma(P) \text{ and } G \models_{reg} \sigma(P')) \text{ or } (G \models_{reg} \sigma(P) \\
&\qquad \text{and } \forall \sigma'; G \models_{reg} \sigma'(P'), \sigma \perp \sigma') \\
&\Leftrightarrow G \models_{reg} \sigma(P) \text{ and } [G \models_{reg} \sigma(P') \text{ or } \forall \sigma'; G \models_{reg} \sigma'(P'), \sigma \perp \sigma'] \\
&\Leftrightarrow G \models_{reg} \sigma(P \texttt{ OPT } P') \\
\sigma \in \mathcal{S}_{reg}(P \texttt{ FILTER } K, G) &\Leftrightarrow \sigma \in \mathcal{S}_{reg}(P, G) \text{ and } \sigma(K) = \top \\
&\Leftrightarrow G \models_{reg} \sigma(P) \text{ and } \sigma(K) = \top \\
&\Leftrightarrow G \models_{reg} \sigma(P \texttt{ FILTER } K)
\end{aligned}
$$

□

This shows the equivalence between the semantic definition (Definition 26) and the algebraic definition (Definition 29)

5.4 Entailment regimes

The good news with the semantic definition is that SPARQL answers are defined semantically, independently of any evaluation mechanism. It is also defined with respect to a simple entailment relation. Hence it becomes possible to obtain a different query language by changing this entailment relation.

The SPARQL specification [PRUD'HOMMEAUX and SEABORNE 2008] introduces the notion of entailment regimes. These are currently further refined by W3C [GLIMM and OGBUJI 2013] (however, still under the definition of "graph pattern homomorphisms" instead of a semantic definition). An entailment regime defines more than the change of entailment relation. It covers:

Graph language (G): which types of graphs are allowed (e.g., RDF, RDFS, OWL);
Query language (q): which kinds of queries are allowed (e.g., restriction or extension of the constraints);
Entailment relation ($\models$): the entailment relation used for defining the answers (e.g., RDF-entailment, RDFS-entailment, which OWL flavour);
Answer type (σ): the σ which will be taken as answer (e.g., are blanks allowed, skolemisation of blank nodes, does the solution lead to a consistent theory);

Syntax error handling : what has to be done in case q is defined incorrectly (e.g., raise an exception, ignore the incorrect part);

Inconsistency handling : what has to be done in case G is inconsistent (e.g., raising an exception, answering a particular subset of the answers).

We have defined SPARQL semantics so that it is easy to change the entailment regime.

5.5 Conclusion

SPARQL is a base query language which mainly allows for querying RDF graphs. As we have seem, it is possible to provide a semantic description of the language based on RDF semantics.

We show below how the query language can be changed (§6) and how the entailment relation may be changed for querying graphs modulo ontologies (§7).

5.6 Exercises

Exercise 18 (SPARQL queries)**.** *Given the following SPARQL query:*

```
SELECT ?t
PREFIX
  foaf: http://xmlns.com/foaf/0.1/
  m: http://mydomain.com/myExample#
  dc: http://purl.org/dc/elements/1.1/
  rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#
WHERE {
  ?x foaf:name "Albert".
  ?x ?r ?l.
  ?r rdf:type rdf:Property.
  ?l rdf:type m:Roman.
  ?l dc:title ?t.
}
```

1. *What is the informal meaning of this query?*
2. *Draw the RDF graph corresponding to the graph patterns of the query.*
3. *What is the difference between such graph patterns and simple RDF graphs?*
4. *Evaluate this query on each of the well-founded graphs of Figure 2.6 of Exercise 1 (p26) and provide the answer.*
5. *What must be added to this query for returning the year of publication of the* `m:Roman` ***if it is available****?*

Exercise 19 (Simple SPARQL query)**.**

1. *Can you express a SPARQL query returning all* `o:TonicPackage` *as defined in the OWL axiom of question 3 of Exercise 17?*

Exercise 20 (SPARQL and queries)**.** *Given the following SPARQL query:*

```
SELECT ?m, ?n
PREFIX  foaf: http://xmlns.com/foaf/0.1/
WHERE {
  ?x foaf:mbox ?m.
  ?x foaf:knows ?y.
  ?z foaf:mbox "keith@i.com".
  ?z marriedWith ?y.
  OPTIONAL { ?x foaf:name ?n }
}
```

1. *What is, informally, the meaning of this query (tell it in English or French)?*
2. *Draw the GRDF graph corresponding to the graph pattern of this query.*
3. *Evaluate the query on the graphs of Figure 2.7 of Exercise 2 (p27) and provide the results.*
4. *Evaluate this query on the closure obtained at Question 4 of Exercise 9 and provide the results.*

Exercise 21 (SPARQL query containment)**.** *Consider the following queries q_1 and q_2 on the RDF graph G of Exercise 3:*

$q_1 =$ `SELECT` $?w$ `FROM` G `WHERE` $(\langle ?w$ `o:wrote` $?x\rangle$ `AND` $\langle ?x$ `rdf:type o:Poem`$\rangle)$ `UNION` $\langle ?w$ `o:translated` $?x\rangle$

$q_2 =$ `SELECT` $?w$ `FROM` G `WHERE` $(\langle ?w$ `o:wrote` $?l\rangle$ `UNION` $\langle ?w$ `o:translated` $?l\rangle)$ `AND` $\langle ?l$ `rdf:type o:Poem`$\rangle$

1. *In the course, we defined the distinguished variables $\vec{B}$, the queried graph G and the query pattern P. Identify them in q_1 and q_2.*
2. *Provide the answers of q_1 and q_2 with respect to the graph G.*

Query containment $q \sqsubseteq q'$ between two queries $q =$ `SELECT` $\vec{B}$ `FROM` G `WHERE` P and $q' =$ `SELECT` $\vec{B}$ `FROM` G `WHERE` P' is defined by the fact that for any RDF graph, the answers to q are included in those to q' ($\forall G, \mathcal{A}(\vec{B}, G, P) \subseteq \mathcal{A}(\vec{B}, G, P')$).

3. *What does the answer to the previous questions tell you about query containment between q_1 and q_2?*
4. *Do you think that query containment holds in some direction between q_1 and q_2 (either $q_1 \sqsubseteq q_2$ or $q_2 \sqsubseteq q_1$)?*
5. *Provide a proof for this. This may be done semantically by using the interpretation of query patterns or syntactically by translating queries into logic and showing that the query containment statement is a theorem.*

Chapter 6

Extending SPARQL

SPARQL is a basic query language based on triple. However, for many applications, more elaborate used of queries are necessary. So, different work have attempted to improve the language in some directions.

There can be different approaches for extending, or reducing, SPARQL:

- adding operators (`AND`, `OR`, etc.), like `COUNT(.)` or `MINUS` or,
- changing the definition of basic graph patterns, like having path expressions instead of triples.

The extensions mentioned as example above have been introduced in SPARQL 1.1 [HARRIS and SEABORNE 2013].

We consider below several such extensions in the perspective of taking ontologies into account when querying data.

6.1 NSPARQL

We offer here a slightly different use of these expressions that we name NSPARQL for differentiating it from the original nSPARQL [PÉREZ et al. 2010]. However, the merits of the approach are directly inherited from the original nSPARQL. Our presentation differs on two points:

- We use a simplified, but equivalent, description of nested regular expressions;
- NSPARQL is simply SPARQL using nested regular expressions in predicate position of graph patterns while nSPARQL does not consider the projection operator (SELECT) [PÉREZ et al. 2010].

6.1.1 nSPARQL syntax

Definition 30 (Regular expression)**.** *A* regular expression *is an expression built from the following grammar:*

$$re ::= axis \mid axis::a \mid re \mid re/re \mid re|re \mid re^*$$

with $a \in \mathcal{U}$ *and* $axis \in \{\texttt{self}, \texttt{next}, \texttt{next}^{-1}, \texttt{edge}, \texttt{edge}^{-1}, \texttt{node}, \texttt{node}^{-1}\}$.

In the following, we use the positive closure of a path expression R denoted by R^+ and defined as $R^+ = R/R^*$.

Regarding the precedence among the regular expression operators, it is as follows: *, /, then |. Parentheses may be used for breaking precedence rules.

The model underlying nSPARQL is that of XPath which navigates within XML structures. Hence, the axis denotes the type of node object which is selected at each step, respectively, the current node (`self` or $\texttt{self}^{-1}$), the nodes reachable through an outbound triple (`next`), the nodes that can reach the current node

through an incident triple (`next`$^{-1}$), the properties of outbound triples (`edge`), the properties of incident triples (`edge`$^{-1}$), the object of a predicate (`node`) and the predicate of an object (`node`$^{-1}$). This is illustrated by Figure 6.1.

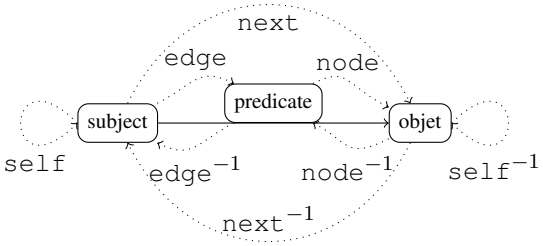


Figure 6.1: nSPARQL axes.

Definition 31 (Nested regular expression)**.** *A* nested regular expression *is an expression built from the following grammar (with $a \in \mathcal{U}$):*

$$nre ::= axis \mid axis::a \mid axis::[nre] \mid nre \mid nre/nre \mid nre|nre \mid nre^*$$

Contrary to simple regular expressions, nested regular expressions may constrain nodes to satisfy additional secondary paths.

Nested regular expressions are used in triple patterns in predicate position, to define nSPARQL triple patterns.

Definition 32 (nSPARQL triple pattern)**.** *An* nSPARQL triple pattern *is a triple $\langle s, p, o\rangle$ such that $s \in \mathcal{T}$, $o \in \mathcal{T}$ and p is a nested regular expression.*

Example 18 (nSPARQL triple pattern)**.** *Assume that one wants to retrieve the pairs of cities such that there is a way of traveling by any transportation mean. The following nSPARQL pattern expresses this query:*

$$P = \langle ?city_1, (next :: [(next :: sp)^*/self :: transport])^+, ?city_2\rangle$$

This pattern expresses a sequence of properties such that each property (predicate) is a sub-property of the property "transport".

Example 19 (nSPARQL triple pattern)**.** *In the context of molecular biology, nSPARQL expressions may be very useful. For instance, part of the graph patterns used for the query of Example 14 can be expressed by:*

```
?x next::rn:inhibits / next::rn:regulates ?z
```

which finds all pairs of nodes such that the first one inhibits a regulator of the second one. It can be further enhanced by using transitive closure:

```
?x next::rn:inhibits / next::rn:regulates+ ?z
```

expressing that we want a path between two nodes going through a first inhibition and then an arbitrary non null number of regulatory steps (+ is the usual notation such that `a+` corresponds to `a/a*`). Nested expressions allow for going further by constraining any step in the path. So,

```
?x next::rn:inhibits[ next::rdf:type/self::dm:gap ] / next::rn:regulates+ ?z
```

requires, in addition, the second node in the path to be a gap gene.

From nSPARQL triple patterns, it is also possible to create a query language. As in SPARQL, a set of nSPARQL triple patterns is called an nSPARQL basic graph pattern. nSPARQL graph patterns may be defined in the usual way, i.e., by replacing triple patterns by nSPARQL triple patterns.

Definition 33 (nSPARQL graph pattern)**.** *An* nSPARQL graph pattern *is defined inductively by:*

- *every nSPARQL triple pattern is an nSPARQL graph pattern;*
- *if P_1 and P_2 are two nSPARQL graph patterns and K is a SPARQL constraint, then (P_1 AND P_2), (P_1 UNION P_2), (P_1 OPT P_2), and (P_1 FILTER K) are nSPARQL graph patterns.*

For time complexity reasons the designers of the nSPARQL language choose to define a more restricted language than SPARQL [PÉREZ et al. 2010]. Contrary to SPARQL queries, nSPARQL queries are reduced to nSPARQL graph patterns, constructed from nSPARQL triple patterns, plus SPARQL operators AND, UNION, FILTER, and OPT. They do not allow for the projection operator (SELECT). This prevents, when checking answers, that uncontrolled variables have to be evaluated.

6.1.2 nSPARQL semantics

In order to define the semantics of nSPARQL, we need to know the semantics of nested regular expressions [PÉREZ et al. 2010]. Here we depart from the semantics given in [PÉREZ et al. 2010] by adding a third variable in the interpretation whose sole purpose is to compact the set of rules. Both definitions are equivalent.

Definition 34 (Nested path interpretation)**.** *Given a nested path p and an RDF graph G, the interpretation of p in G (denoted $[\![p]\!]_G$) is defined by:*

$$
\begin{aligned}
[\![\texttt{self}]\!]_G &= \{\langle x,x,x\rangle; x \in \mathcal{T}\}\\
[\![\texttt{self}]\!]_G &= \{\langle x,x,x\rangle; x \in \mathcal{T}\}\\
[\![\texttt{next}]\!]_G &= \{\langle x,y,z\rangle; \exists z; \langle x,z,y\rangle \in G\}\\
[\![\texttt{edge}]\!]_G &= \{\langle x,z,z\rangle; \exists z; \langle x,z,y\rangle \in G\}\\
[\![\texttt{node}]\!]_G &= \{\langle z,y,z\rangle; \exists z; \langle x,z,y\rangle \in G\}\\
[\![nre\texttt{::}a]\!]_G &= \{\langle x,y,a\rangle \in [\![nre]\!]_G\}\\
[\![nre^{-1}]\!]_G &= \{\langle y,x,z\rangle; \langle x,y,z\rangle \in [\![nre]\!]_G\}\\
[\![nre_1\texttt{[}nre_2\texttt{]}]\!]_G &= \{\langle x,y,z\rangle \in [\![nre_1]\!]_G; \exists w,k; \langle z,w,k\rangle \in [\![nre_2]\!]_G\}\\
[\![nre_1/nre_2]\!]_G &= \{\langle x,y,z\rangle; \langle x,w,k\rangle \in [\![nre_1]\!]_G \& \langle w,y,z\rangle \in [\![nre_2]\!]_G\}\\
[\![nre_1\texttt{|}nre_2]\!]_G &= [\![nre_1]\!]_G \cup [\![nre_2]\!]_G\\
[\![nre\texttt{*}]\!]_G &= [\![\texttt{self}]\!]_G \cup [\![nre]\!]_G \cup [\![nre/nre]\!]_G \cup [\![nre/nre/nre]\!]_G \cup \ldots
\end{aligned}
$$

The evaluation of a nested regular expression R over an RDF graph G is defined as the sets of pairs $\langle a,b\rangle$ of nodes in G, such that b is reachable from a in G by following a path that conforms to R. We write $\langle x,y\rangle \in [\![R]\!]_G$ as a shortcut for $\exists z$ such that $\langle x,y,z\rangle \in [\![R]\!]_G$.

Definition 35 (Satisfaction of a nSPARQL triple pattern)**.** *Given a basic nested path graph pattern $\langle s,p,o\rangle$ and an RDF graph G, $\langle s,o\rangle$ satisfies p in G (denoted $G \models_{nSPARQL} \langle s,p,o\rangle$) if and only if $\exists\sigma$; $\langle\sigma(s),\sigma(o)\rangle \in [\![p]\!]_G$*

This nested regular expression evaluation problem is solved efficiently through an effective procedure provided in [PÉREZ et al. 2010].

Theorem 10 (Complexity of nested regular expression evaluation [PÉREZ et al. 2010])**.** *The evaluation problem for a nested regular expression R over an RDF graph G can be solved in time $O(|G| \times |R|)$.*

Answers to nSPARQL queries follow the same definition as for SPARQL (Definition 26) but with maps satisfying nSPARQL triple patterns.

Definition 36 (Evaluation of a nSPARQL triple pattern)**.** *The evaluation of a nSPARQL triple pattern $\langle x, R, y\rangle$ over an RDF graph G is:*

$$[\![\langle x, R, y\rangle]\!]_G = \{\sigma | dom(\sigma) = \{x, y\} \cap \mathcal{B} \text{ and } \langle \sigma(x), \sigma(y)\rangle \in [\![R]\!]_G\}$$

[PÉREZ et al. 2010] shows that avoiding the projection operator (SELECT), keeps the complexity of nSPARQL *basic*, i.e., conjunctive, graph pattern evaluation to polynomial and mentions that adding projection would make it NP-complete.

Clearly, nSPARQL is a good navigational language, but there still are useful queries that could not be expressed. For example, it cannot be used to find nodes connected with transportation mean that is not a bus or transportation means belonging to Air France, i.e., containing the IRI of the company.

6.1.3 NSPARQL

It is also possible to create a query language from nSPARQL triple patterns by simply replacing SPARQL triple patterns by nSPARQL triple patterns. We call such a language NSPARQL for differentiating it from the original nSPARQL [PÉREZ et al. 2010]. However, the merits of the approach are directly inherited from the original nSPARQL.

A NSPARQL query for the select form is `SELECT` $\vec{B}$ `FROM` u `WHERE` P such that P is a nSPARQL graph pattern (see Definition 33). Hence, NSPARQL graph patterns are built on top of nSPARQL in the same way as SPARQL is built on top of GRDF and PSPARQL is built on top of PRDF.

Answers to NSPARQL queries are based on the extension of $\models_{nSPARQL}$ nSPARQL graph patterns following Definition 26 using $\models_{nSPARQL}$ as the entailment relation.

Definition 37 (Answers to an NSPARQL query)**.** *Let* `SELECT` $\vec{B}$ `FROM` u `WHERE` P *be a NSPARQL query with P a nSPARQL graph pattern and G be the (G)RDF graph identified by the IRI u, then the set of* answers *to this query is:*

$$\mathcal{A}^o(\vec{B}, G, P) = \{\sigma|^{\vec{B}}_{\vec{B}} |\ G \models_{nSPARQL} \sigma(P)\}.$$

The complexity of NSPARQL query evaluation is likely to be PSPACE-complete as that of SPARQL (see §5.2).

6.2 PSPARQL

Both presented approaches have their drawbacks. Independently, we have designed a third approach which does not suffer from the same problems.

In order to address the problems raised by querying RDF graphs modulo RDF Schemas, we first present the PSPARQL query language for RDF which we introduced in [ALKHATEEB, BAGET, et al. 2009]. Unlike SPARQL, PSPARQL can express queries with regular path expressions instead of relations and variables on the edges. For instance, it allows for finding all maternal genes regulated by the `dm:bcd` gene through an arbitrary sequence of inhibitions.

In the next section, we show how the additional expressive power provided by PSPARQL can be used for answering queries modulo RDF Schema.

The added expressiveness of PSPARQL is achieved by extending SPARQL graph patterns, hence any SPARQL query is a PSPARQL query. SPARQL graph patterns, based on GRDF graphs, are replaced by PRDF graphs that are introduced below through their syntax (§6.2.1) and semantics (§6.2.2). They are then naturally replaced within the PSPARQL context (§6.2.3).

6.2.1 PRDF syntax

Let Σ be an alphabet. A *language* over Σ is a subset of Σ^*: its elements are sequences of elements of Σ called *words*. A (non empty) word $\langle a_1, \ldots, a_k \rangle$ is denoted $a_1 \cdot \ldots \cdot a_k$. If $A = a_1 \cdot \ldots \cdot a_k$ and $B = b_1 \cdot \ldots \cdot b_q$ are two words over Σ, then $A \cdot B$ is the word over Σ defined by $A \cdot B = a_1 \cdot \ldots \cdot a_k \cdot b_1 \cdot \ldots \cdot b_q$.

Definition 38 (Regular expression pattern)**.** *Let Σ be an alphabet, X be a set of variables, the set $\mathcal{R}(\Sigma, X)$ of* regular expression patterns *is inductively defined by:*

- $\forall a \in \Sigma$, $a \in \mathcal{R}(\Sigma, X)$ *and* $!a \in \mathcal{R}(\Sigma, X)$*;*
- $\forall x \in X$, $x \in \mathcal{R}(\Sigma, X)$*;*
- $\epsilon \in \mathcal{R}(\Sigma, X)$*;*
- *If $A \in \mathcal{R}(\Sigma, X)$ and $B \in \mathcal{R}(\Sigma, X)$ then $A|B$, $A \cdot B$, A^*, A^{-1}, $A^+ \in \mathcal{R}(\Sigma, X)$.*

A regular expression over $(\mathcal{U}, \mathcal{B})$ can be used to define a language over the alphabet made of $\mathcal{U} \cup \mathcal{B}$. PRDF graphs are GRDF graphs where predicates in the triples are regular expression patterns constructed over the set of URIs and the set of variables.

Definition 39 (PRDF graph)**.** *A* PRDF triple *is an element of $\mathcal{U} \cup \mathcal{B} \times \mathcal{R}(\mathcal{U}, \mathcal{B}) \times \mathcal{T}$. A PRDF graph is a set of PRDF triples.*

Hence all GRDF graphs are PRDF graphs.

Example 20 (PRDF Graph)**.** *PRDF graphs can express interesting features of regulatory networks. For instance, one may observe that* `dm:bcd` *promotes* `dm:Kr` *without knowing if this action is direct or indirect. Hence, this can be expressed by* `dm:bcd rn:promotes+ dm:Kr`*.*

A generalized version of the graph pattern of the query of Example 14 can be expressed by:

```
?x rn:inhibits.rn:regulates* ?z.
?x rn:promotes+ ?z.
?x rdf:type rn:gene.
```

6.2.2 PRDF semantics

To be able to define models of PRDF graphs, we have first to express path semantics within RDF semantics to support regular expressions.

Definition 40 (Support of a regular expression)**.** *Let $I = \langle I_R, I_P, I_{EXT}, \iota \rangle$ be an interpretation of a vocabulary $V = \mathcal{U} \cup \mathcal{L}$, ι' be an extension of ι to $B \subseteq \mathcal{B}$, and $R \in \mathcal{R}(\mathcal{U}, B)$, a pair $\langle x, y \rangle$ of $(I_R \times I_R)$* supports *R in ι' if and only if one of the two following conditions are satisfied:*

(i) the empty word $\epsilon \in L^*(R)$ and $x = y$;
(ii) there exists a word of length $n \geq 1$ $w = w_1 \cdot \ldots \cdot w_n$ where $w \in L^*(R)$ and a sequence of resources of I_R $x = r_0, \ldots, r_n = y$ such that $\langle r_{i-1}, r_i \rangle \in I_{EXT}(\iota'(w_i))$, $1 \leq i \leq n$.

Instead of considering paths in RDF graphs, this definition considers paths in the interpretations of PRDF graphs, i.e., paths are now relating resources. This is used in the following definition of PRDF models in which it replaces the direct correspondences that exist in RDF between a relation and its interpretation, by a correspondence between a regular expression and a sequence of relation interpretations. This allows for matching regular expressions, e.g., r^+, with variable length paths.

Definition 41 (Model of a PRDF graph)**.** *Let G be a PRDF graph, and $I = \langle I_R, I_P, I_{EXT}, \iota \rangle$ be an interpretation of a vocabulary $V \supseteq \mathcal{V}(G)$, I is a* PRDF model *of G if and only if there exists an extension ι' of ι to $\mathcal{B}(G)$ such that for every triple $\langle s, R, o \rangle \in G$, $\langle \iota'(s), \iota'(o) \rangle$ supports R in ι'.*

This definition extends the definition of RDF models, and they are equivalent when all regular expression patterns R are reduced to atomic terms, i.e., IRIs or variables. PRDF entailment is defined as usual:

Definition 42 (PRDF entailment)**.** *Let P and G be two PRDF graphs, G* PRDF-entails *P (noted $G \models_{PRDF} P$) if and only if all models of G are models of P.*

It is possible to define the interpretation of a regular expression evaluation as those pairs of resources which support the expression in all models.

Definition 43 (Regular expression interpretation)**.** *The interpretation $[\![R]\!]_G$ of a regular expression R in a PRDF graph G is the set of nodes which satisfy the regular expression in all models of the graph:*

$$[\![R]\!]_G = \{\langle x, y \rangle | \, \forall I \textit{ PRDF model of } G, \langle x, y \rangle \textit{ supports } R \textit{ in } I\}$$

It is thus possible to formulate the regular expression evaluation problem:
Problem: Regular expression evaluation
Input: An RDF graph G, a regular expression R, and a pair $\langle a, b \rangle$
Question: Does $\langle a, b \rangle \in [\![R]\!]_G$?

6.2.3 PSPARQL

PSPARQL is an extension of SPARQL introducing the use of paths in SPARQL graph patterns. PSPARQL graph patterns are built on top of PRDF in the same way as SPARQL is built on top of GRDF.

Definition 44 (PSPARQL graph pattern)**.** *A* PSPARQL graph pattern *is defined inductively by:*

- *every PRDF graph is a PSPARQL graph pattern;*
- *if P_1 and P_2 are two PSPARQL graph patterns and K is a SPARQL constraint, then (P_1 `AND` P_2), (P_1 `UNION` P_2), (P_1 `OPT` P_2), and (P_1 `FILTER` K) are PSPARQL graph patterns.*

A PSPARQL query for the select form is `SELECT` $\vec{B}$ `FROM` u `WHERE` P such that P is a PSPARQL graph pattern.

Analogously to SPARQL, the set of answers to a PSPARQL query is defined inductively from the set of maps of the PRDF graphs of the query into the RDF knowledge base. The definition of an answer to a PSPARQL query is thus identical to Definition 26, but it uses PRDF entailment.

Definition 45 (Answers to a PSPARQL query)**.** *Let `SELECT` $\vec{B}$ `FROM` u `WHERE` P be a PSPARQL query with P a PRDF graph pattern, and G be the RDF graph identified by the IRI u, then the set of* answers *to this query is:*

$$\mathcal{A}^\star(\vec{B}, G, P) = \{\sigma|^{\vec{B}}_{\vec{B}} | G \models_{PRDF} \sigma(P)\}$$

6.3 cpSPARQL and CPSPARQL

CPSPARQL has been defined for addressing two main issues. The first one comes from the need to extend PSPARQL and thus to allow for expressing constraints on nodes of traversed paths; while the second one comes from the need to answer PSPARQL queries modulo RDFS so that the transformation rules could be applied to PSPARQL queries [ALKHATEEB 2008].

In addition to CPSPARQL, we present cpSPARQL [ALKHATEEB and EUZENAT 2014], a language using CPSPARQL graph patterns in the same way as nSPARQL does.

6.3.1 CPSPARQL syntax

The notation that we use for the syntax of CPSPARQL is slightly different from the one defined in the original proposal [ALKHATEEB 2008]. The original one uses `edge` and `node` constraints to express constraints on predicates (or edges) and nodes of RDF graphs, respectively. In this paper, we adopt the `axes` borrowed from XPath, with which the reader may be more familiar, as done for nSPARQL. This also will allow us to better compare cpSPARQL and nSPARQL. Additionally, in the original proposal, `ALL` and `EXISTS` keywords are used to express constraints on all traversed nodes or to check the existence of a node in the traversed path that satisfies the given constraint. We do not use these keywords in the fragment presented below since they do not add expressiveness with respect to RDFS semantics, i.e., the fragment still captures RDFS semantics.

Constraints act as filters for paths that must be traversed by constrained regular expressions and select those whose nodes satisfy encountered constraint.

Definition 46 (Constrained regular expression)**.** *A* constrained regular expression *is an expression built from the following grammar:*

$$cre ::= axis \mid axis::a \mid axis::[?x : \psi] \mid axis::]?x : \psi[\mid cre \mid cre/cre \mid cre|cre \mid cre^*$$

with ψ a set of triples belonging to $\mathcal{U} \cup \mathcal{B} \cup \{?x\} \times cre \times \mathcal{T} \cup \{?x\}$ and FILTER-expressions over $\mathcal{B} \cup \{?x\}$. ψ is called a CPRDF-constraint and $?x$ its head variable.

Constrained regular expressions allow for constraining the item in one axis to satisfy a particular constraint, i.e., to satisfy a particular graph pattern (here an RDF graph) or filter. We introduce the closed square brackets and open square brackets notation for distinguishing between constraints which export their variable (it may be assigned by the map) and constraints which do not export it (the variable is only notational). This is equivalent to the initial CPSPARQL formulation, in which the variable was always exported, since CPSPARQL can ignore such variables through projection.

We use $\mathcal{B}(R)$ for the set of variables occurring as the head variable of an open bracket constraint in R.

Constraint nesting is allowed because constrained regular expressions may be used in the graph pattern of another constrained regular expression as in Example 21.

Example 21 (Constrained regular expression)**.** *The following constrained regular expression could be used to find nodes connected by transportation means that are not buses:*

$$(next :: [?p : \{\langle ?p, (next :: sp)^*, transport\rangle FILTER(?p! = bus)\}])^+$$

In contrast to nested regular expressions, constrained regular expressions can apply constrains (such as SPARQL constraints) in addition to simple nested path constraints.

Constrained regular expressions are used in triple patterns, precisely in predicate position, to define CPSPARQL.

Definition 47 (CPSPARQL triple pattern)**.** *A* CPSPARQL triple pattern *is a triple $\langle s, p, o\rangle$ such that $s \in \mathcal{T}$, $o \in \mathcal{T}$ and p is a constrained regular expression.*

Definition 48 (CPSPARQL graph pattern)**.** *A* CPSPARQL graph pattern *is defined inductively by:*

- *every CPSPARQL triple pattern is a CPSPARQL graph pattern;*
- *if P_1 and P_2 are two CPSPARQL graph patterns and K is a SPARQL constraint, then (P_1 `AND` P_2), (P_1 `UNION` P_2), (P_1 `OPT` P_2), and (P_1 `FILTER` K) are CPSPARQL graph patterns.*

Example 22 (CPSPARQL graph pattern)**.** *The following CPSPARQL graph pattern could be used to retrieve the set of pairs of cities connected by a sequence of transportation means (which are not buses) such that one city in France and the other one in Jordan:*

$$\{\langle ?city_1, (next :: [?p : \{\langle ?p, (next :: sp)^*, transport\rangle FILTER(?p! = bus)\}])^+, ?city_2\rangle$$
$$\langle ?city_1, next :: cityIn, France\rangle$$
$$\langle ?city_2, next :: cityIn, Jordan\rangle\}$$

If open square brackets were used, this graph pattern would, in addition, bind the $?p$ variable to a matching value, i.e., the transportation means used.

By restricting CPRDF constraints, it is possible to define a far less expressive language. cpSPARQL is such a language.

Definition 49 (cpSPARQL regular expression [ALKHATEEB and EUZENAT 2014])**.** *A* cpSPARQL regular expression *is an expression built from the following grammar:*

$$\begin{aligned} cpre ::= \ & axis \mid axis::a \mid axis::]?x : TRUE[\\ & \mid axis::[?x : \{\langle ?x, cpre, v\rangle\}\{FILTER(?x)\}] \\ & \mid cpre \mid cpre/cpre \mid cpre|cpre \mid cpre^* \end{aligned}$$

such that v is either a distinct variable $?y$ or a constant (an element of $U \cup L$) and $FILTER(?x)$ is the usual SPARQL filter condition containing at most the variable $?x$ and v if v is a variable.

The first specific form, with open square brackets, has been preserved so that cpSPARQL triples cover SPARQL basic graph patterns, i.e., allow for variables in predicate position. In the other specific forms, a cpSPARQL constraint is either a cpSPARQL regular expression containing $?x$ as the only variable and/or a SPARQL FILTER constraint. Hence, such a regular expression may have several constraints, but each constraint can only expose one variable and it cannot refer to variables defined elsewhere. It is clear that any cpSPARQL regular expression is a constrained regular expression.

Deciding if a CPSPARQL triple is a cpSPARQL triple can be decided in linear time in the size of the regular expression used.

Example 23 (cpSPARQL triple patterns)**.** *The query of Example 18 could be expressed by the following cpSPARQL pattern:*

$$\langle ?city_1, (next :: [?p : \{\langle ?p, (next :: sp)^*, transport\rangle\}])^+, ?city_2\rangle$$

The constraint $\psi = ?p : \{\langle ?p, (next :: sp)^*, transport\rangle\}$ is used to restrict the properties (in this pattern the constraint is applied to properties since the axis `next` is used) to be only a transportation mean.

Example 21 provides another cpSPARQL regular expression. By contrast, CPSPARQL graph patterns allow for queries like:

$$\begin{aligned} next :: [?p; \{\langle ?p, (next :: sp)*, ?z\rangle, \\ & \langle ?q, (next :: sp)*, ?z\rangle, \\ & \langle ?p, owl : inverseOf, ?q\rangle, \\ & FILTER(regex(?z, iata.org)\}] \end{aligned}$$

which is not a cpSPARQL regular expression since it uses more than two variables.

It is possible to develop languages based on cpSPARQL regular expressions following what is done with constrained regular expressions.

6.3.2 CPSPARQL semantics

Intuitively, a constrained regular expression $next::[\psi]$ (where $\psi = ?p : \{\langle ?p, sp^*, transport\rangle\}$) is equivalent to $next::p$ if p satisfies the constraint ψ, i.e., p should be a sub-property of $transport$ (when p is substituted to the variable $?p$).

Definition 50 (Satisfied constraint in an RDF graph)**.** *Let G be an RDF graph, s and o be two nodes of G and $\psi = x:C$ be a constraint, then s and o satisfies ψ in G (denoted $\langle s, o\rangle \in [\![\psi]\!]_G$) if and only if one of the following conditions is satisfied:*

1. *C is a triple pattern $C = \langle x, R, y\rangle$, and $\langle x^x_s, y^y_o\rangle \in [\![R^x_s]\!]_G$, where K^z_r means that r is substituted to the variable z if $K = z$ or K contains the variable z. If z is a constant then $z = r$.*
2. *C is a SPARQL filter constraint and $C^{x,y}_{s,o} = \top$, where $C^{x,y}_{s,o} = \top$ means that the constraint obtained by the substitution of s to each occurrence of the variable x and o to each occurrence of the variable y in C is evaluated to true*[1]*.*
3. *$C = P\ FILTER\ K$, then 1 and 2 should be satisfied*

As for nested regular expressions, the evaluation of a constrained regular expression R over an RDF graph G is defined as a binary relation $[\![R]\!]_G$, by a pair of nodes $\langle a, b\rangle$ such that a is reachable from b in G by following a path that conforms to R. The following definition extends Definition 34 to take into account the semantics of terms with constraints.

Definition 51 (Constrained path interpretation)**.** *Given a constrained regular expression P and an RDF graph G, if P is unconstrained then the interpretation of P in G (denoted $[\![P]\!]_G$) is as in Definition 34, otherwise the interpretation of P in G is defined as:*

$$
\begin{aligned}
[\![self :: [\psi]]\!]_G &= \{\langle x, x\rangle |\ \exists z; x \in voc(G) \wedge \langle x, z\rangle \in [\![\psi]\!]_G\} \\
[\![next :: [\psi]]\!]_G &= \{\langle x, y\rangle |\ \exists z, w; \langle x, z, y\rangle \in G \wedge \langle z, w\rangle \in [\![\psi]\!]_G\} \\
[\![edge :: [\psi]]\!]_G &= \{\langle x, y\rangle |\ \exists z, w; \langle x, y, z\rangle \in G \wedge \langle z, w\rangle \in [\![\psi]\!]_G\} \\
[\![node :: [\psi]]\!]_G &= \{\langle x, y\rangle |\ \exists z, w; \langle z, x, y\rangle \in G \wedge \langle z, w\rangle \in [\![\psi]\!]_G\} \\
[\![axis^{-1} :: [\psi]]\!]_G &= \{\langle x, y\rangle |\ \langle y, x\rangle \in [\![axis :: [\psi]]\!]_G\}
\end{aligned}
$$

Definition 52 (Answer to a CPSPARQL triple pattern)**.** *The evaluation of a CPSPARQL triple pattern $\langle x, R, y\rangle$ over an RDF graph G is defined as the following set of maps:*

$$[\![\langle x, R, y\rangle]\!]_G = \{\sigma \mid dom(\sigma) = \{x, y\} \cap \mathcal{B} \cup \mathcal{B}(R) \text{ and } \langle \sigma(x), \sigma(y)\rangle \in [\![\sigma(R)]\!]_G\}$$

such that $\sigma(R)$ is the constrained regular expression obtained by substituting the variable $?x$ appearing in a constraint with open brackets in R by $\sigma(?x)$.

This semantics also applies to cpSPARQL graph patterns.

6.3.3 Evaluating cpSPARQL regular expressions

In order to establish the complexity of cpSPARQL we follow [PÉREZ et al. 2010] to store an RDF graph as an adjacency list: every $u \in voc(G)$ is associated with a list of pairs $\alpha(u)$. For instance, if $\langle s, p, o\rangle \in G$, then $\langle \texttt{next}::p, o\rangle \in \alpha(s)$ and $\langle \texttt{edge}^{-1}::o, s\rangle \in \alpha(p)$. Also, $\langle \texttt{self}::u, u\rangle \in \alpha(u)$, for $u \in voc(G)$. The set of terms of a constrained regular expression R, denoted by $\mathcal{T}(R)$, is constructed as follows:

[1] Except for the case of bound (see Definition 29 and the discussion after it).

$$\begin{aligned}\mathcal{T}(R) =& \{R\} \text{if } R \text{ is either } \texttt{axis}, \texttt{axis}::\text{a, or } \texttt{axis}::\psi \\ \mathcal{T}(R_1/R_2) =& \mathcal{T}(R_1|R_2) = \mathcal{T}(R_1) \cup \mathcal{T}(R_2) \\ \mathcal{T}(R_1^*) =& \mathcal{T}(R_1)\end{aligned}$$

Let $\mathcal{A}_R = (Q, \mathcal{T}(R), s_0, F, \delta)$ be the $\epsilon - NFA$ of R constructed in the usual way using the terms $\mathcal{T}(R)$, where $\delta : Q \times (\mathcal{T}(R) \cup \{\epsilon\}) \to 2^Q$ be its transition function. In the evaluation algorithm, we use the product automaton $G \times \mathcal{A}_R$ (in which $\delta' : \langle voc(G) \times Q\rangle \times (\mathcal{T}(R) \cup \{\epsilon\}) \to 2^{voc(G)\times Q}$ is its transition function). We construct $G \times \mathcal{A}_R$ as follows:

- $\langle u, q\rangle \in voc(G) \times Q$, for every $u \in voc(G)$ and $q \in Q$;
- $\langle v, q\rangle \in \delta'(\langle u, p\rangle, s)$ iff $q \in \delta(p, s)$; and one of the following conditions satisfied:
  - $s = \texttt{axis}$ and there exists a s.t. $\langle \texttt{axis}::a, v\rangle \in \alpha(u)$
  - $s = \texttt{axis}::a$ and $\langle \texttt{axis}::a, v\rangle \in \alpha(u)$
  - $s = \texttt{axis}::\psi$ and there exists b s.t. $\langle \texttt{axis}::b, v\rangle \in \alpha(u)$ and $b \in [\![\psi]\!]_G$

Algorithm 2 (Eval) solves the evaluation problem for a constrained regular expression R over an RDF graph G. This algorithm is almost the same as the one in [PÉREZ et al. 2010] which solves the evaluation problem for nested regular expressions R over an RDF graph G. The Eval algorithm calls the Algorithm 1 (LABEL), which is an adaptation of the LABEL algorithm of [PÉREZ et al. 2010] in which we modify only the first two steps.

Algorithm 1 LABEL(G, exp):

1. **for each** $axis :: [\psi] \in D_0(exp)$ **do**
2. call Label(G, exp') //where $exp' = exp1/self :: p$ if $\psi = ?x :< ?x, exp1, p >$; $exp' = exp1/self :: p$ if $\psi = ?x :< ?x, exp1, ?y >$
3. construct A_{exp}, and assume that q_0 is its initial state and F is its set of final states
4. construct $G \times A_{exp}$
5. **for each** state (u, q_0) that is connected to a state (v, q_f) in $G \times A_{exp}$, with $q_f \in F$ **do**
6. $label(u) := label(u) \cup exp$

The algorithm as the same $O(|G| \times |R|)$ time complexity as usual regular expressions [YANNAKAKIS 1990; MENDELZON and WOOD 1995] and nested regular expressions [PÉREZ et al. 2010] evaluation.

Algorithm 2 Eval$(G, R, \langle a, b\rangle)$

Data: An RDF graph G, a constrained regular expression R, and a pair $\langle a, b\rangle$.
Result: YES if $\langle a, b\rangle \in [\![R]\!]_G$; otherwise NO.

for each $u \in voc(G)$ **do**
  $label(u) := \varnothing$
LABEL (G, R)
construct $\mathcal{A}_R$ (assume q_0 : initial state and F : set of final states)
construct the product automaton $G \times \mathcal{A}_R$
if a state $\langle b, q_f\rangle$ with $q_f \in F$, is reachable from $\langle a, q_0\rangle$ in $G \times \mathcal{A}_R$ **then**
  return YES;
else
  return NO;
end if

Theorem 11 (Complexity of cpSPARQL regular expression evaluation)**.** *Eval solves the evaluation problem for constrained regular expression in time* $O(|G| \times |R|)$*.*

6.3.4 Conclusion

We presented three different SPARQL extensions based on regular expressions. They offer a navigational view of RDF querying contrasting with the filtering view of SPARQL. Similar extensions are now integrated within the definition of SPARQL 1.1 [HARRIS and SEABORNE 2013].

These are justified on their own, as well as by the used that will be done of them for evaluating queries module RDFS semantics (§7.2). A comparison of these extensions can be found in [ALKHATEEB and EUZENAT 2013] which can be summarised in Figure 6.2.

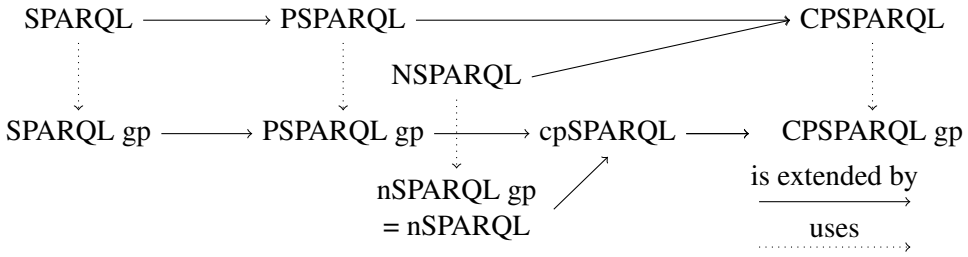


Figure 6.2: Query languages and their graph patterns.

Streams and navigation

Many applications in sensor networks, smart cities or the web of things are generating streams of RDF triples or quads, i.e., in this context, RDF triples labelled by their occurrence time. One may want to query such streams for detecting particular events occurring and alerting some monitor about it. Because the expressiveness of triples is very poor, these queries rely on graph patterns. The difficulty is that applications should be able to deal quickly with large amounts of real time data. For that purpose, they can take advantage of techniques defined in artificial intelligence for rule compilation (RETE, TREAT), as well as more sophisticated techniques for recognising chronicles [DOUSSON et al. 1993]. This has to be combined with time windowing techniques which allows to match events only related to closer events.

The C-SPARQL language has been designed for that purpose [BARBIERI et al. 2010] and theoretical studies for stream reasoning are under development [BECK et al. 2015].

The path facilities of SPARQL 1.1 are a good way to specify navigational queries and could allow evaluating queries against the semantic web in a '*follow your nose*' fashion, i.e., by dereferencing reachable IRIs and querying their own data sources. This querying mode is well adapted to the web of data.

However, SPARQL queries are usually evaluated against one particular graph specified in the FROM or GRAPH clauses. The new federated query feature also requires that graph patterns be evaluated against one specific server. Thus the SPARQL recommendation does not specify the semantics of evaluating queries against the web. Moreover, the availability of the web is unpredictable and the evaluation would require, in principle, to look-up a potentially infinite number of data sources. To better specify the semantics of such queries evaluated over the web, *reachability-based* and *context-based query semantics* have been defined [HARTIG and PIRRÓ 2016]. They characterise the set of answer to a query by answers with respect to reachable data sources or those selected from the IRIs instantiating the paths. They also identify *web-safe queries* which can be evaluated according to the new semantics and return complete answers.

Chapter 7

Querying modulo ontologies

The definition of SPARQL through graph homomorphisms does not make it easy to apply it with other semantics than that of simple RDF graphs. Indeed, without correspondence results between consequence and homomorphisms for such languages, query evaluation is disconnected from their semantics. This happens if one queries RDF graphs without taking into account their RDFS Schema or OWL ontologies. SPARQL could have been grounded on the notion of logical consequences instead of graph homomorphism [ALKHATEEB, BAGET, et al. 2009]. Such a foundation would allow to take advantage of any ontology language using model theory for expressing its semantics.

Recently, SPARQL 1.1 introduced the notion of SPARQL entailment regime which replaces graph homomorphism by logical entailment and introduces further constraints on the types of allowed query pattern, expected answers, or what to do in case of inconsistency (see §5.4).

Before that, various attempts have been made to deal with this situation. For instance, it is possible to answer a SPARQL query with respect to an ontology written in ρDF or DL-Lite through simple rewriting of the query as a PSPARQL [ALKHATEEB, BAGET, et al. 2008], SPARQL 1.1 or SQL query. Similar results have been obtained by restricting SPARQL itself [PÉREZ et al. 2010]. Similarly, the SPARQL-DL language [SIRIN and PARSIA 2007] allows for evaluating a fragment of SPARQL (conjunctive queries with limited use of variables) with respect to an OWL DL ontology. Independently from SPARQL, other works have extended Datalog to deal with queries written in DL-Lite$_\mathcal{R}$ (hence in ρDF) [CALI et al. 2009] or studied the complexity of such queries when introducing the SPARQL disjunction operator [ARTALE et al. 2009].

Our goal in this chapter is to consider how to answer SPARQL queries through taking into account the semantics of the RDF graph as a RDFS graph, i.e., by considering that answers are evaluated modulo an ontology. We will attempt at doing so by extending SPARQL.

[MUÑOZ et al. 2009] has introduced the reflexive relaxed semantics for RDFS in which `rdfs:subClassOf` and `rdfs:subPropertyOf` do not have to be reflexive. The entailment relation with this semantics is noted $\models^{\text{nrx}}_{\text{RDFS}}$.

The reflexive relaxed semantics does not change much RDFS. Indeed, from the standard (reflexive) semantics, we can deduce that any class (respectively, property) is a subclass (respectively, subproperty) of itself. For instance, $\langle$`dm:hb rn:inhibits dm:kni`$\rangle$ only entails $\langle$`rn:inhibits sp rn:inhibits`$\rangle$ and variations of this triple in which occurrences of `rn:inhibits` are replaced by variables. The reflexivity requirement only entails reflectivity assertions which do not interact with other triples unless constraints are added to the `rdfs:subPropertyOf` or `rdfs:subClassOf` predicates. Therefore, it is assumed that elements of the RDFS vocabulary appear only in the predicate position. We call *genuine*, RDFS graphs which do not constrain the elements of the ρdf vocabulary (and thus these two predicates), and restrict ourselves to querying genuine RDFS graphs.

However, when issuing queries involving these relations, e.g., with a graph pattern like $\langle$`?x sp ?y`$\rangle$, all properties in the graph will be answers. Since this would clutter results, we assume, as done in [MUÑOZ

et al. 2009], that queries use the reflexive relaxed semantics. It is easy to recover the standard semantics by providing the additional triples when `sp` or `sc` are queried.

By contrast to entailment regimes (§5.4), we concentrate here on the definition of answers which, in particular, replace simple RDF entailment for answering queries. It is possible to define answers to SPARQL queries modulo RDF Schema, by using RDFS entailment as the entailment regime.

Definition 53 (Answers to a SPARQL query modulo RDF Schema)**.** *Let* `SELECT` $\vec{B}$ `FROM` u `WHERE` P *be a SPARQL query with P a GRDF graph and G be the RDFS graph identified by the IRI u, then the set of* answers *to this query* modulo RDF Schema *is:*

$$\mathcal{A}^{\#}(\vec{B}, G, P) = \{\sigma|^{\vec{B}}_{\vec{B}} | G \models^{nrx}_{RDFS} \sigma(P)\}$$

This definition is justified by the analogy between RDF entailment and RDFS entailment in the definition of answers to queries (see [ALKHATEEB, BAGET, et al. 2009]). It does not fully correspond to the RDFS entailment regime defined in [GLIMM and OGBUJI 2013] since we do not restrict the answer set to be finite. This is not strictly necessary, however, the same restrictions as in [GLIMM and OGBUJI 2013] can be applied.

The problem is to specify a query engine that can find such answers. They will be illustrated by querying the graph of Example 2 modulo the RDFS ontology of Example 24.

Example 24 (RDFS)**.** *The RDF graph of Example 2 can be expressed in the context of an RDF Schema which provides more information about the vocabulary that it uses. It specifies subtypes of genes and subtypes of regulation relations.*

```
dm:maternal rdfs:subClassOf rn:gene.
dm:gap rdfs:subClassOf rn:gene.

dm:kni rdf:type dm:gap.
dm:hb rdf:type dm:gap.
dm:Kr rdf:type dm:gap.
dm:tll rdf:type dm:gap.
dm:bcd rdf:type dm:maternal.
dm:cad rdf:type dm:maternal.

rn:regulates rdfs:domain rn:gene.
rn:regulates rdfs:range rn:gene.
rn:inhibits rdfs:subPropertyOf rn:regulates.
rn:promotes rdfs:subPropertyOf rn:regulates.
rn:inhibits_translation rdfs:subPropertyOf rn:inhibits.
rn:inhibits_transcription rdfs:subPropertyOf rn:inhibits.
```

7.1 RDF(S) closure and query answering

One possible approach for querying an RDFS graph G in a sound and complete way is by computing the closure graph of G, i.e., the graph obtained by saturating G with all information that can be deduced using a set of predefined rules called RDFS rules, before evaluating the query over the closure graph (see §3.1.3).

Example 25 (RDFS Closure)**.** *The RDFS closure of the RDF graph of Example 2 augmented by the RDFS triples of Example 24 contains, in particular, the following assertions:*

```
dm:bcd rn:inhibits dm:cad. // [RDFS 9]
dm:hb rn:regulates dm:kni. // [RDFS 9]
dm:hb type rn:gene.        // [RDFS 6]
```

Since queries must adopt the reflexive relaxed semantics, we have to further restrict this closure. It can be obtained by suppressing constraints RDFS8a and RDFS12a from the closure operation. We denote the partial non reflexive closure $\hat{G}\backslash\backslash H$.

Proposition 12 (Completeness of partial non reflexive RDFS closure [ALKHATEEB and EUZENAT 2013])**.** *Let G be a satisfiable genuine RDFS graph and H an RDFS graph, then $G \models^{nrx}_{RDFS} H$ if and only if $(\hat{G}\backslash\backslash H) \models_{RDF} H$.*

This has the following corollary:

Corollary 13.

$$\mathcal{A}^{\#}(\vec{B}, G, P) = \mathcal{A}(\vec{B}, \hat{G}\backslash\backslash P, P)$$

This shows the correctness and completeness of the closure approach.

Example 26 (SPARQL evaluation modulo RDFS)**.** *If the query of Example 14 is evaluated against the RDFS closure of Example 25, it will return the three expected answers:*

$$\{\langle \texttt{dm:hb}, \texttt{dm:kni}, \texttt{dm:Kr}\rangle$$
$$\langle \texttt{dm:bcd}, \texttt{dm:tll}, \texttt{dm:Kr}\rangle$$
$$\langle \texttt{dm:bcd}, \texttt{dm:cad}, \texttt{dm:kni}\rangle\}$$

This can be obtained easily from the simple graph of Example 2 augmented by the triples of Example 25.

With this result, for the query evaluation problem, it is sufficient to enumerate the set of homomorphisms from the query graph pattern(s) into the closure graph.

This approach has several drawbacks which limit its use: It still tends to generate a very large graph which makes it not very convenient, especially if the transformation has to be made on the fly, i.e., when the query is evaluated; It takes time proportional to $|H| \times |G|^2$ in the worst case [MUÑOZ et al. 2009]; Moreover, it is not applicable if one has no access to the graph to be queried but only to a SPARQL endpoint. In this case, it is not possible to compute the closure graph.

Since the complexity of the partial closure has been shown to be polynomial [TER HORST 2005], $\mathcal{A}^{\#}$-ANSWER CHECKING remains PSPACE-complete.

Proposition 14 (Complexity of $\mathcal{A}^{\#}$-ANSWER CHECKING)**.** $\mathcal{A}^{\#}$-ANSWER CHECKING *is PSPACE-complete.*

7.2 Ontologies expansion as path navigation

Another possible approach [MUÑOZ et al. 2009], consists of searching paths between RDF(S) vocabularies at query evaluation time. This results in a simple query language nSPARQL based on nested regular expressions for navigating the RDF graph.

This may be achieved through NSPARQL, PSPARQL or CPSPARQL.

7.2.1 The nSPARQL approach

Definition 54. *The evaluation of an nSPARQL triple pattern $\langle x, R, y\rangle$ over an RDF graph G modulo RDFS is defined as*

$$[\![\langle x, R, y\rangle]\!]^{rdfs}_G = [\![\langle x, R, y\rangle]\!]_{closure(G)}$$

Definition 55 (Answers to an nSPARQL basic graph pattern modulo RDFS)**.** *Let P be a basic nSPARQL graph pattern and G be an RDF graph, then the set of* answers *to P over G modulo RDFS is:*

$$\mathcal{S}^o(P,G) = \bigcap_{t \in P} [\![t]\!]_G^{rdfs}$$

As presented in [PÉREZ et al. 2010], nSPARQL can evaluate queries with respect to RDFS by transforming the queries with rules [MUÑOZ et al. 2009]:

$$\begin{aligned}
\phi(sc) &= (\texttt{next::sc})+ \\
\phi(sp) &= (\texttt{next::sp})+ \\
\phi(dom) &= \texttt{next::dom} \\
\phi(range) &= \texttt{next::range} \\
\phi(type) &= \texttt{next::type/next::sc}* \\
&\quad |\texttt{edge/next::sp}*\texttt{/next::dom/next::sc}* \\
&\quad |\texttt{node}^{-1}\texttt{/next::sp}*\texttt{/next::range} \\
&\quad /\texttt{next::sc}* \\
\phi(p) &= \texttt{next[(next::sp)}*\texttt{/self::p]} \quad (p \notin \{\texttt{sp}, \texttt{sc}, \texttt{type}, \texttt{dom}, \texttt{range}\})
\end{aligned}$$

Example 27 (nSPARQL evaluation modulo RDFS)**.** *The following nSPARQL graph pattern could be used as a query to retrieve the set of pairs of cities connected by a sequence of transportation means such that one city is from France and the other city is from Jordan:*

$$\begin{aligned}
\{&\langle ?city_1, (next::transport)^+, ?city_2\rangle, \\
&\langle ?city_1, next::cityIn, France\rangle, \\
&\langle ?city_2, next::cityIn, Jordan\rangle\}
\end{aligned}$$

When evaluating this graph pattern against the RDF graph of Figure 2.1, it returns the empty set since there is no explicit "transport" property between the two queried cities. However, considering the RDFS semantics, it should return the following set of pairs:

$$\{\langle ?city_1 \leftarrow Paris, ?city_2 \leftarrow Amman\rangle, \langle ?city_1 \leftarrow Grenoble, ?city_2 \leftarrow Amman\rangle\}$$

To answer the above graph pattern considering RDFS semantics, it could be transformed to the following nSPARQL graph pattern:

$$\begin{aligned}
\{&\langle ?city_1, (next::[(next::sp)^*/self::transport])^+, ?city_2\rangle, \\
&\langle ?city_1, next::cityIn, France\rangle, \\
&\langle ?city_2, next::cityIn, Jordan\rangle\}
\end{aligned}$$

This encoding is correct and complete with respect to RDFS entailment.

Theorem 15 (Completeness of ϕ on triples [PÉREZ et al. 2010] (Theorem 3))**.** *Let $\langle x, p, y\rangle$ be a SPARQL triple pattern with $x, y \in (\mathcal{U} \cup \mathcal{B})$ and $p \in \mathcal{U}$, then for any RDF graph G:*

$$[\![\langle x, p, y\rangle]\!]_G^{rdfs} = [\![\langle x, \phi(p), y\rangle]\!]_G$$

This results uses the natural extension of ϕ to nSPARQL graph patterns to answer SPARQL queries modulo RDFS.

Proposition 16 (Completeness of ϕ on graph patterns [PÉREZ et al. 2010])**.** *Given a basic graph pattern P and an RDFS graph G,*

$$G \models^{nrx}_{RDFS} P \text{ iff } G \models_{nSPARQL} \phi(P)$$

These results about nSPARQL can be transferred to NSPARQL query answering:

Proposition 17 (Answers to a SPARQL query modulo RDFS by NSPARQL [ALKHATEEB and EUZENAT 2013])**.** *Let `SELECT` $\vec{B}$ `FROM` u `WHERE` P be a SPARQL query with P a SPARQL graph pattern and G the RDFS graph identified by the IRI u, then the set of answers to this query is:*

$$\mathcal{A}^{\#}(\vec{B}, G, P) = \{\sigma|^{\vec{B}}_{\vec{B}} \mid \sigma \in \mathcal{S}^o(\phi(P), G)\}$$

Example 28 (NSPARQL Query)**.** *The result of transforming the query of Example 14 with ϕ is:*

```
SELECT ?x, ?y, ?z
FROM ...
WHERE {
  ?x next::(next::sp)*/self::rn:inhibits] ?y.
  ?y next::(next::sp)*/self::rn:regulates] ?z.
  ?x next::(next::sp)*/self::rn:promotes] ?z.
  ?x next::rdf:type/next::sc*
    |edge/next::sp*/next::dom/next::sc*
    |node-1/next::sp*/next::range/next::sc* rn:gene.
}
```

This query provides the correct set of answers for the RDF graph of Example 2 modulo the RDF Schema of Example 24 (given in Example 26).

The main problem with NSPARQL is that, because nested regular expressions do not contain variables, it does not preserve SPARQL queries. Indeed, it is impossible to express a query containing the simple triple $\langle ?x\ ?y\ ?z \rangle$. This may seem like a minor problem, but in fact NSPARQL graph patterns prohibits dependencies between two variables as freely as it is possible to express in SPARQL.

7.2.2 Answering SPARQL queries modulo RDFS through PSPARQL

To overcome the limitations of previous approaches when querying RDF graphs modulo an RDF Schema, we provide a new approach which rewrites a SPARQL query into a PSPARQL query using a set of rules, and then evaluates the transformed query over the graph to be queried. In particular, we show that every SPARQL query Q to evaluate over an RDFS graph G can be transformed into a PSPARQL query $\tau(Q)$ such that evaluating Q over $\hat{G}$, the closure graph of G, is equivalent to evaluating $\tau(Q)$ over G.

The query rewriting approach is similar in spirit to the query rewriting methods using a set of views [PAPAKONSTANTINOU and VASSALOS 1999; CALVANESE, DE GIACOMO, LENZERINI, et al. 2000; GRAHNE and THOMO 2003]. In contrast to these methods, our approach uses the data contained in the graph, i.e., the rules are inferred from RDFS entailment rules. We define a rewriting function τ from RDF graph patterns to PRDF graph patterns through a set of rewriting rules over triples (which naturally extends to basic graph patterns and queries). $\tau(Q)$ is obtained from Q by applying the possible rule(s) to each triple in Q.

Definition 56 (Basic RDFS graph pattern expansion)**.** *Given an RDF triple t, the* RDFS expansion *of t is a*

finite PSPARQL graph pattern $\tau(t)$ defined as:

$$\begin{aligned}
\tau(\langle s, \mathtt{sc}, o\rangle) =& \{\langle s, \mathtt{sc}^+, o\rangle\}\\
\tau(\langle s, \mathtt{sp}, o\rangle) =& \{\langle s, \mathtt{sp}^+, o\rangle\}\\
\tau(\langle s, p, o\rangle) =& \{\langle s, ?x, o\rangle, \langle ?x, \mathtt{sp}^*, p\rangle\} (p \notin \{\mathtt{sc}, \mathtt{sp}, \mathtt{type}\})\\
\tau(\langle s, \mathtt{type}, o\rangle) =& \{\langle s, \mathtt{type} \cdot \mathtt{sc}^*, o\rangle\}\\
& \mathtt{UNION} \{\langle s, ?p, ?y\rangle, \langle ?p, \mathtt{sp}^* \cdot \mathtt{dom} \cdot \mathtt{sc}^*, o\rangle\}\\
& \mathtt{UNION} \{\langle ?y, ?p, s\rangle, \langle ?p, \mathtt{sp}^* \cdot \mathtt{range} \cdot \mathtt{sc}^*, o\rangle\}
\end{aligned}$$

The first rule handles the transitive semantics of the subclass relation. Finding the subclasses of a given class can be achieved by navigating all its direct subclasses. The second rule handles similarly the transitive semantics of the subproperty relation. The third rule tells that the subject-object pairs occurring in the subproperties of a given property are inherited to it. Finally, the fourth rule expresses that the instance mapped to s has for type the class mapped to o (we use the word "mapped" since s and/or o can be variables) if one of the following conditions holds:

1. the instance mapped to s has for type one of the subclasses of the class mapped to o by following the subclass relationship zero or several times. The zero times is used since s can be directly of type o;
2. there exists a property of which s is subject and such that the instances appearing as a subject must have for type one of the subclasses of the class mapped to o;
3. there exists a property of which s is object and such that the instances appearing as an object must have for type one of the subclasses of the class mapped to o.

The latter rule takes advantage of a feature of PSPARQL which does not exist in NSPARQL: the ability to have variables in predicates.

Example 29 (PSPARQL Query)**.** *The result of transforming the query of Example 14 with τ is:*

```
SELECT ?x, ?y, ?z
FROM ...
WHERE {
  ?x ?r ?y. ?r sp* rn:inhibits.
  ?y ?t ?z.?t sp* rn:regulates.
  ?x ?s ?z. ?s sp* rn:promotes.
  ( ?x rdf:type.sc* rn:gene.
  UNION
   { ?x ?u ?v. ?u sp*.dom.sc* rn:gene.}
  UNION
   { ?v ?u ?x. ?u sp*.range.sc* rn:gene.}
  )
}
```

This query provides the correct set of answers for the RDF graph of Example 2 modulo the RDF Schema of Example 24 (given in Example 26).

Any SPARQL query can be answered modulo an RDF Schema by rewriting the resulting query in PSPARQL and evaluating the PSPARQL query.

Proposition 18 (Answers to a SPARQL query modulo RDF Schema by PSPARQL [ALKHATEEB and EUZENAT 2013])**.**

$$\mathcal{A}^{\#}(\vec{B}, G, P) = \mathcal{A}^{\star}(\vec{B}, G, \tau(P))$$

This transformation does not increase the complexity of PSPARQL which is the same as the one of SPARQL:

Proposition 19 (Complexity of $\mathcal{A}^\star$-ANSWER CHECKING [ALKHATEEB and EUZENAT 2013])**.** $\mathcal{A}^\star$-ANSWER CHECKING *is PSPACE-complete.*

7.2.3 SPARQL queries modulo RDFS with CPSPARQL

Like for nSPARQL, constraints allow for encoding RDF Schemas within queries.

Definition 57 (RDFS triple pattern expansion [ALKHATEEB 2008])**.** *Given an RDF triple t, the* RDFS expansion *of t, denoted by $\tau(t)$, is defined as:*

$$\begin{aligned}
\tau(\langle s, \texttt{sc}, o\rangle) =& \langle s, \texttt{next::sc}^+, o\rangle \\
\tau(\langle s, \texttt{sp}, o\rangle) =& \langle s, \texttt{next::sp}^+, o\rangle \\
\tau(\langle s, \texttt{dom}, o\rangle) =& \langle s, \texttt{next::dom}, o\rangle \\
\tau(\langle s, \texttt{range}, o\rangle) =& \langle s, \texttt{next::range}, o\rangle \\
\tau(\langle s, \texttt{type}, o\rangle) =& \langle s, \texttt{next::type/next::sc}^*| \\
& \texttt{edge/(next::sp)}^*\texttt{/next::dom/(next::sc)}^*| \\
& \texttt{node}^{-1}\texttt{/(next::sp)}^*\texttt{/next::range/(next::sc)}^*, o\rangle \\
\tau(\langle s, p, o\rangle) =& \langle s, (\texttt{next::}[?x : \{\langle ?x, (\texttt{next::sp})^*, p\rangle\}]), o\rangle \\
& p \notin \{\texttt{sp}, \texttt{sc}, \texttt{type}, \texttt{dom}, \texttt{range}\}
\end{aligned}$$

The RDFS expansion of an RDF triple is a cpSPARQL triple.

The extra variable $?x$ introduced in the last item of the transformation, is only used inside the constraint of the constrained regular expression and so it is not considered to be in $dom(\sigma)$, i.e., only variables occurring as a subject or an object in a CPSPARQL triple pattern are considered in maps (see Definition 52). Therefore, the projection operator (SELECT) is not needed to restrict the results of the transformed triple as in the case of PSPARQL [ALKHATEEB, BAGET, et al. 2009], as illustrated in the following example.

Example 30 (SPARQL query transformation)**.** *Consider the following SPARQL query that searches pairs of nodes connected with a property p*

```
SELECT ?X ?Y
WHERE ?X p ?Y .
```

It is possible to answer this query modulo RDFS by transforming this query into the following PSPARQL query:

```
SELECT ?X ?Y
WHERE ?X ?P ?Y . ?P sp* p .
```

The evaluation of the above PSPARQL query is the map $\{?X \leftarrow a, ?P \leftarrow b, ?Y \leftarrow c\}$. So, to actually obtain the desired result, a projection (SELECT) operator must be performed since an extra variable $?P$ is used in the transformation. It is argued in [PÉREZ et al. 2010] that including the projection (SELECT) operator to the conjunctive fragment of PSPARQL makes the evaluation problem NP-hard.

On the other hand, the query could be answered by transforming it, with the τ function of Definition 57, to the following cpSPARQL query (in which there is no need for the projection operator):

```
?X next::[?z:  ?z (next::sp)* p ] ?Y
```

Since the variable $?z$ is used inside the constraint, the answer to this query will be $\{?X \leftarrow a, ?Y \leftarrow b\}$ (see Definition 51).

This has the important consequence that any nSPARQL graph pattern can be translated in a cpSPARQL graph pattern with similar structure and no additional variable. Hence, no additional projection operation (SELECT) is required for answering nSPARQL queries in cpSPARQL.

Theorem 20 ([ALKHATEEB and EUZENAT 2013])**.** *Let $\langle x, p, y\rangle$ be a SPARQL triple pattern with $x, y \in (\mathcal{U} \cup \mathcal{B})$ and $p \in \mathcal{U}$, then $[\![\langle x, p, y\rangle]\!]_G^{rdfs} = [\![\langle x, \tau(p), y\rangle]\!]_G$ for any RDF graph G.*

7.3 Querying modulo DL-lite

DL-Lite (§3.2.5) allows for efficient query answering through rewriting the query.

In this case, queries are restricted to disjunctions of conjunctive atoms. This means the SPARQL queries must be `UNION` of basic graph patterns.

The result is obtained by using the DL-Lite terminology in order to transform the query into a set (union) of transformed queries that can be evaluated independently. It also requires that the ontology be satisfiable.

See Marie-Christine Rousset slides for more details.

7.4 Conclusion

Properly answering queries when ontologies are used for describing data is very important. We presented here approaches for doing so specific to SPARQL and RDF schema. Concerning, RDFS queries, research is very active to find the best way to evaluate these queries. However, these approaches cannot be easily extended to more expressive languages like languages of the OWL family.

7.5 Exercises

Exercise 22 (SPARQL modulo ontologies)**.**

1. *The way queries are answered in Exercise 20 is not the standard way for answering queries modulo DL-Lite ontologies. What is the standard method? Rewrite the query of Exercise 20 with this method and give its results.*

Exercise 23 (Query modulo ontology)**.** *We now consider the ontology o of Exercise 11 and the following queries:*

- $q_3 =$ `SELECT` $?y$ `FROM` o `WHERE` $\langle ?x, \texttt{o:translated}, ?y\rangle$;
- $q_4 =$ `SELECT` $?y$ `FROM` o `WHERE` $\langle ?y, \texttt{rdf:type}, \texttt{o:Literature}\rangle$.

1. *Do you think that query containment holds in some direction between q_3 and q_4 (either $q_3 \sqsubseteq q_4$ or $q_4 \sqsubseteq q_3$)? Tell why.*
2. *Can you provide a definition for query containment modulo an ontology o ($q \sqsubseteq_o q'$)?*
3. *Does it return different answers for q_3 and q_4?(either $q_3 \sqsubseteq_o q_4$ or $q_4 \sqsubseteq_o q_3$)? Tell why.*

Exercise 24 (SPARQL queries)**.** *Consider the following statements added to the statements of Exercise 12 (p.55):*

```
ex:MyTeam rdf:type ex:ModernTeam2 .
ex:Kay ex:member ex:MyTeam .
ex:Kay rdf:type ex:Man .
ex:Jo ex:member ex:MyTeam .
```

1. *If one queries this graph with* `SELECT ?x WHERE ?x ex:member ex:MyTeam . ?x rdf:type ex:Woman .`*, what would be the answer?*
2. *What would be necessary for* `ex:Jo` *to be an answer?*

Exercise 25 (Refactoring graphs with SPARQL CONSTRUCT)**.**

1. *Both movies and books may be considered as work expressions. Write a SPARQL query able to return all such items in graph G of Exercise 4 with their title and, if possible, the place they are stored in.*
2. *Consider that, instead of extending the schema, one would prefer to refactor the graph G of Exercise 4 so that it corresponds to the schema S of Exercise 14. Create a SPARQL CONSTRUCT query able to extract the data from G and generate a graph complying to S (check on the example above).*
3. *Considering that you have a SPARQL engine able to answer queries taking into account RDFS semantics, write the same query as in Question 1 using the schema S and the answer to Question 1.*

Exercise 26 (Computing closure with SPARQL CONSTRUCT)**.** *It is possible to generate RDF graphs from an RDF graph through the use of SPARQL CONSTRUCT. Consider the query Q:*

CONSTRUCT { *?y* `rdf:type` *?x* }
WHERE { *?u* `rdfs:subClassOf` *?x* . *?y* `rdf:type` *?u* }.

1. *Apply Q to the graph G of Figure 2.8 (p. 29) and give the resulting graph. Let us call the result $Q(G)$, does $G \cup Q(G) \models_{RDF} P'$?*
2. *For any map σ, does*

   $$\sigma(?u)\ \texttt{rdfs:subClassOf}\ \sigma(?x).\ \sigma(?y)\ \texttt{rdf:type}\ \sigma(?u) \models_{RDFS} \sigma(?y)\ \texttt{rdf:type}\ \sigma(?x)$$

   (explain why)
3. *Is this related with the rules of the ter Horst closure? (tell if it corresponds more closely to one of these rules).*
4. *Would it be possible to transform all closure rules as SPARQL CONSTRUCTS? Do it or explain why.*
5. *What more is needed for computing the closure?*

Part III

Networks of ontologies

Chapter 8

Alignments and networks of ontologies

The semantic web relies on knowledge deployed and connected over the web. This knowledge is based on ontologies expressed in languages such as RDF, RDF Schema and OWL. Because of the multiplicity of ontologies, they may be connected through alignments expressing correspondences between their concepts. This allows for translating assertions across ontologies or merging them. This can be seen as a network of ontologies related by alignments.

The goal of this third part is to provide a formal account of such networks of ontologies.

8.1 Motivation

We are dealing with a space of knowledge sources called the semantic web. Each source provides knowledge expressed in its own vocabulary called ontology[1]. There is no a priori reason that this knowledge be expressed:

- in the same knowledge representation language,
- with the same vocabulary, or
- with the same modelling of that vocabulary.

However, this heterogeneity should not prevent from interpreting this knowledge, like the fact that people use different natural languages and may have different state of mind does not fully prevent them to understand each others.

Example 31 (Heterogeneity)**.** *Consider two knowledge sources to be found on the web The first one, o, is a simple set of triples :*

```
ex:May ex:childOf ex:Peter.
ex:Peter ex:childOf ex:Paul.
ex:Paul ex:childOf ex:Zeno.
```

The second source, o', is an axiom in some first order logic dialect:

```
ex:grandParent( x, y ) ⇐ ∃ z; ex:parent( x, z ) ∧ ex:parent( z, y )
```

One would expect, given these two sources to be able to deduce that `ex:grandParent( ex:Paul, ex:May )`*, that we will refer to as δ, holds. This is however not the case since:*

- *The two sources use different knowledge representation languages (syntactic heterogeneity);*
- *The two sources use different vocabularies:* `child` *versus* `parent` *(terminological heterogeneity);*

[1] In fact, an ontology provides both the vocabulary and axioms governing the terms in the vocabulary.

– *Even if the vocabulary were the same, the modelling of these notion may be different.*

This can be summarised by the following remarks:

– $o \not\models \delta$ *and* $o' \not\models \delta$*;*
– $o \cup o'$ *impossible*
– *even* $o \cup o' \not\models \delta$

This problem can be solved by imposing to the word a unique ontology and a unique knowledge representation language. However, this is both unrealistic and non wishable. Instead, we propose to draw relations between knowledge sources mostly based on their ontologies. We call "ontology matching" the process of finding the relations between ontologies and we call "alignment" the result of this process expressing declaratively these relations [EUZENAT and SHVAIKO 2013]. In an open world in which ontologies evolve, managing ontologies requires using alignments for expressing the relations between ontologies.

Example 32 (Alignment)**.** *In the case of Example 31, we can imagine to provide an alignment* A *containing the following correspondence:*

$$\langle \texttt{ex:childOf}^{-1}, \equiv, \texttt{parent} \rangle$$

This alignment would solve our current problem because $o \cup o' \cup A \models \delta$*. However, the merge of the ontologies provided by* $o \cup o' \cup A$ *may not be the solution that works in any case. For instance, if* o' *contained in addition, the following axioms:*

```
ex:parent( x, y ) ⇒ ex:Human( x ) ∧ ex:Human( y )
ex:childOf( x, y ) ⇒ ex:Tree( x ) ∧ ex:Tree( y )
ex:Human ⊥ ex:Tree
```

The merge would be an inconsistent knowledge source. In such a case, it may be better to use the alignment as a data transformation program that would import whatever `ex:childOf` *formula entailed by* o *into the corresponding* `ex:parent` *formula in* o'*.*

Several classes of applications can be considered (they are more extensively described in [EUZENAT and SHVAIKO 2013], we only summarise them here). They are the following:

Ontology evolution uses matching for finding the changes that have occurred between two ontology versions [DE LEENHEER and MENS 2008].

Schema integration uses matching for integrating the schemas of different databases under a single view;

Catalog integration uses matching for offering an integrated access to on-line catalogs;

Data integration uses matching for integrating the content of different databases under a single database;

P2P information sharing uses matching for finding the relations between ontologies used by different peers [ROUSSET et al. 2006], Figure 8.1 is an example of the use of alignments for mediation in such systems;

Web service composition uses matching between ontologies describing service interfaces in order to compose web services by connecting their interfaces;

Multiagent communication uses matching for finding the relations between the ontologies used by two agents and translating the messages they exchange;

Context matching in ambient computing uses matching of application needs and context information when applications and devices have been developed independently and use different ontologies;

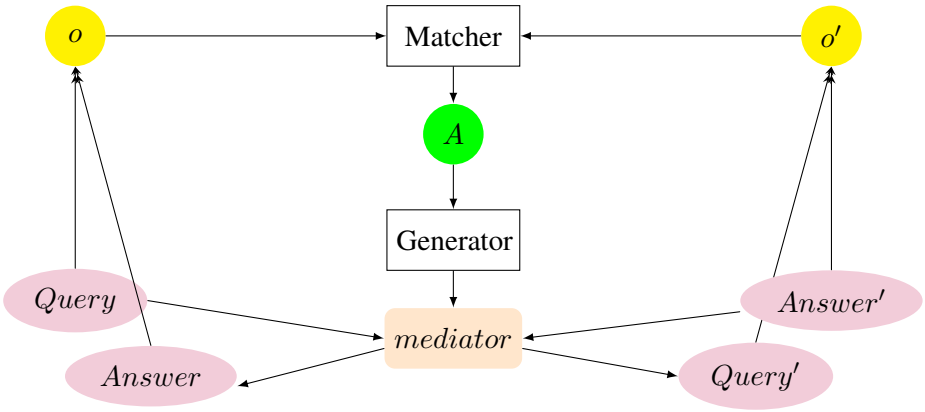


Figure 8.1: Query mediation (from [EUZENAT and SHVAIKO 2013]). From two matched ontologies o and o', resulting in alignment A, a $mediator$ is generated. This allows the transformation of queries expressed with the entities of the first ontology into a query using the corresponding entities of a matched ontology and the translation back of the results from the second ontology to the first one.

Query answering uses ontology matching for translating user queries about the web;

Semantic web browsing uses matching for dynamically (while browsing) annotating web pages with partially overlapping ontologies.

It is clear, from the above examples, that matching ontologies is a major issue in ontology related activities. It is not circumscribed to one area of ontology, but applies to any application that communicates through ontologies.

We consider here, the foundations of ontology matching and alignments within the semantic web. In particular, we will establish the semantics of alignments and the constraints this raises on the interpretation of ontologies.

We first precisely define what are alignments through their syntax (§8.2) and semantics (§10) before introducing networks of ontologies and their semantics. There is no "standard" semantics for networks of ontologies, so we provide here an abstract view that aims at covering all of them and we provide an example with our equalising semantics (§11). Finally, we consider the use of the semantics of networks of ontologies within the context of semantic peer-to-peer systems (§12.1).

Considering an open semantic web requires facing the heterogeneity of data and ontologies. On way of doing so consists of considering a network of ontologies related by alignments expressing the relations between them. Such alignments are used for interpreting the set of ontologies.

We will consider these under the standpoint of their syntax (§8.2 and 9) and their semantics (§10 and 11) Finally, we use this interpretation for evaluating queries in a distributed world according to the semantics (§12).

8.2 Alignment syntax

Alignments express the correspondences between entities belonging to different ontologies[2] . All definitions here are given for matching between two ontologies. In case of multiple matching, the definitions can be straightforwardly extended by using n-ary correspondences. A correspondence must consider the two

[2]This part is mostly written from [EUZENAT and SHVAIKO 2013].

corresponding entities and the relation that is supposed to hold between them. We provide the definition of the alignment following the work in [EUZENAT and SHVAIKO 2013].

Since the related entities are an important part of alignments, they have to be defined. We separate the matched entities from the ontology language because it can be desirable to have a different language for identifying the matched entities. Given an ontology language, we use an *entity language* for expressing those entities that will be put in correspondence by matching. The expressions of this language will depend on the ontology on which expressions are defined.

The entity language can be simply made of all the formulas of the ontology language based on the ontology vocabulary. It can restrict its scope to particular kinds of formulas from the language, for instance, atomic formulas, or even to terms of the language, like class expressions. It can also restrict the entities to be only named entities. This is convenient in the context of the semantic web to restrict entities to those identifiable by their IRIs. The entity language can also be an extension of the ontology language: this can be a query language, such as SPARQL [PRUD'HOMMEAUX and SEABORNE 2008], adding operations for manipulating ontology entities that are not available in the ontology language itself, like concatenating strings or joining relations. Finally, this entity language can combine both extension and restriction, e.g., by authorising any boolean operations over named ontology entities.

Definition 58 (Entity language)**.** *Given an ontology language L, an entity language Q_L is a function from any ontology $o \subseteq L$ which defines the matchable entities of ontology o.*

In the following we will assume that each ontology interpretation can be extended to an interpretation of the entity language associated with the ontology.

The next important component of the alignment is the relation that holds between the entities. We identify a set of relations Θ that is used for expressing the relations between the entities. Matching algorithms primarily use the equivalence relation ($=$) meaning that the matched objects are the same or are equivalent if these are formulas. It is possible to use relations from the ontology language within Θ. For instance, using OWL, it is possible to take advantage of the `owl:equivalentClass`, `owl:disjointWith` or `rdfs:subClassOf` relations in order to relate classes of two ontologies. These relations correspond to set-theoretic relations between classes: *equivalence* ($=$); *disjointness* ($\perp$); *more general* ($\sqsupseteq$). They can be used without reference to any ontology language. Finally, relations can be of any type and are not restricted to relations present within the ontology language, such as fuzzy relations or probability distributions over a complete set of relations, or similarity measures.

A more general view is the introduction of algebras of relations instead of these ad hoc sets of relations (see Chapter 9).

With these ingredients, it is possible to define the correspondences that have to be found by matching algorithms.

Definition 59 (Correspondence)**.** *Given two ontologies o and o' with associated entity languages Q_L and $Q_{L'}$ and a set of alignment relations Θ, a* correspondence *is a triple:*

$$\langle e, e', r \rangle,$$

such that

- $e \in Q_L(o)$ *and* $e' \in Q'_{L'}(o')$*;*
- $r \in \Theta$*.*

The correspondence $\langle e, e', r \rangle$ asserts that the relation r holds between the ontology entities e and e'.

Example 33 (Correspondence)**.** *For example, a simple kind of correspondence is as follows:*

http://book.ontologymatching.org/example/culture-shop.owl#Book $=$
http://book.ontologymatching.org/example/library.owl#Volume

It asserts the equivalence relation with full confidence between what is denoted by two IRIs, namely the Book *class in one ontology and the* Volume *class in another one. Some examples of more complex correspondences are as follows:*

$$\begin{aligned} & & \mathit{Book}(x) \\ \mathit{author}(x, concat(w.\mathit{firstname}, w.\mathit{lastname})) & \Leftarrow_{.85} & \wedge\ \mathit{writtenBy}(x, w) \\ & & \wedge\ \mathit{Writer}(w) \end{aligned}$$

is a Horn clause expressing that if there exists a Book x *written by* Writer w*, the* author *of* x *in the first ontology is identified by the concatenation of the first and last name of* w*. The confidence in this clause is quantified with a .85 degree.*

There can be several possible correspondences for the same entities depending on the language in which correspondences are expressed. For instance, one could have the simple correspondence that speed *in one ontology is equivalent to* velocity *in another one:*

$$\mathit{speed} \equiv \mathit{velocity}$$

or record that they are expressed in miles per hour and metre per second respectively:

$$\begin{aligned} \mathit{speed} &= \mathit{velocity} \times 2.237 \\ 0.447 \times \mathit{speed} &= \mathit{velocity} \end{aligned}$$

For pragmatic reasons, the relationship between two entities is often assigned a degree of confidence which can be viewed as a measure of trust in the fact that the correspondence holds – 'I trust 70% the fact that the correspondence is correct or reliable' – and can be compared with the certainty measures provided with meteorological forecasts. This measure is taken from a confidence structure.

Definition 60 (Confidence structure)**.** *A confidence structure is an ordered set of degrees* $\langle \Xi, \leq \rangle$ *for which there exists a greatest element* $\top$ *and a smallest element* $\bot$*.*

In [EUZENAT and SHVAIKO 2013], confidence was included in Definition 59 (then Definition 2.11) as an additional component. We now consider that it is better to consider confidence as metadata over the correspondence itself.

The usage of confidence degrees is that the higher the degree with regard to $\leq$, the most likely the relation holds. It is convenient to interpret the greatest element as the boolean true and the smallest element as the boolean false.

The most widely used structure is based on the real number unit interval $[0\ 1]$, but some systems simply use the boolean lattice. Some other possible structures are fuzzy degrees, probabilities or other lattices. [GAL et al. 2005] has investigated the structure of fuzzy confidence relations. This structure can be extended, for instance, if one wants to compose alignments. Thus, in this case, it may be necessary to define operations for combining these degrees.

Finally, an alignment is defined as a set of correspondences.

Definition 61 (Alignment)**.** *Given two ontologies* o *and* o'*, an* alignment *is a set of correspondences between pairs of entities belonging to* $Q_L(o)$ *and* $Q_{L'}(o')$ *respectively.*

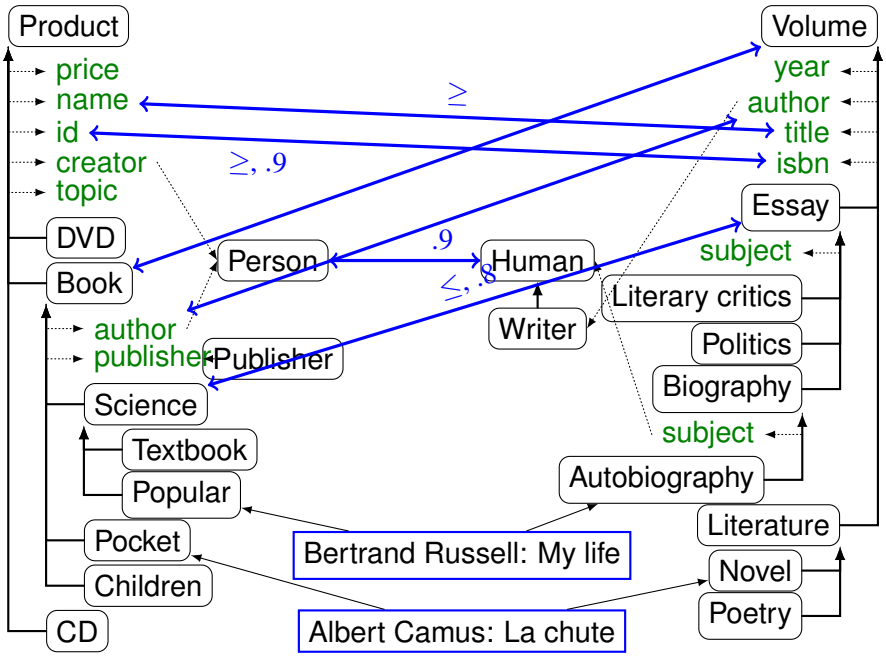


Figure 8.2: Alignment between two ontologies. Correspondences are expressed by arrows. By default their relation is $=$ and their confidence value is 1.0; otherwise, these are mentioned near the arrows.

Example 34 (Alignment)**.** *Figure 8.2 displays a possible alignment for a pair of ontologies. It can be expressed by the following correspondences:*

$$\textit{Book} =_{1.0} \textit{Volume} \qquad \textit{name} \geq_{1.0} \textit{title}$$
$$\textit{id} \geq_{.9} \textit{isbn} \qquad \textit{author} =_{1.0} \textit{author}$$
$$\textit{Person} =_{.9} \textit{Human} \qquad \textit{Science} \leq_{.8} \textit{Essay}$$

In the following, for the sake of simplicity, we only consider that ontologies are description logic terminologies and their entities are defined classes identified by IRIs. We will only consider $=$, $\leq$, $\geq$ and $\perp$ as relations expressing equivalence, subsumption and disjointness between classes. However, results will also apply to other relations.

So far, our alignments are very simple: they are sets of pairs of entities from two ontologies. However, there are, in the literature, at least three types of n:m, multiple or complex alignments:

1. alignments involving more than two ontologies produced by multiple matching, that we may call multialignments,
2. alignments involving correspondences between more than two entities (still belonging to two ontologies),
3. alignments with entities involved in more than one correspondence that are denoted by the use of * (zero-or-more) or + (more-than-zero) in their cardinalities.

In case of multiple matching (1), the alignments must contain correspondences relating more than two entities. The definitions above must then be extended accordingly. This is not covered further here.

The second kind of correspondences (2) can be thought of as using non binary relations. For instance, in schema matching, some authors [SHETH and LARSON 1990; RAHM and BERNSTEIN 2001] tend to consider that a correspondence like:

$$\text{address} = \text{street} + \text{“ ”} + \text{number}$$

is a ternary complex relation ($\cdot = \cdot + \text{“ ”} + \cdot$) between three entities address, street and number. However, given the nature of the problem: matching two ontologies, we will consider that these objects can be grouped by operators in the entity language Q_L which may include operators such as concatenation, arithmetic operations or logical connectors for that purpose. Hence, in our setting, the correspondence above is simply a normal correspondence in which the binary relation is equivalence ($=$) and the ontology entities are address and street+“ ”+number. This is the main reason why we consider ontology entities, the latter entity is a term built on strings and operations on strings (here concatenation $+$). In its simplest expression the only construction can be a set.

Option (3) is related to the multiplicity of the alignment if it is considered as a relation. By analogy with mathematical functions, it is useful to define some properties of the alignments. These apply when the only considered relation is equality ($=$) and the confidence measures are not taken into account. One can ask for a total alignment with regard to one ontology, i.e., all the entities of one ontology must be successfully mapped to the other one. This property is purposeful whenever thoroughly transcribing knowledge from one ontology to another is the goal: there is no entity that cannot be translated.

One can also require the alignment to be injective with regard to one ontology, i.e., all the entities of the other ontology is part of at most one correspondence. Injectivity is useful in ensuring that entities that are distinct in one ontology remain distinct in the other one. In particular, this contributes to the reversibility of alignments.

Definition 62 (Total alignment, injective alignment)**.** *Given two ontologies o and o', an alignment A over o and o' is called a* total alignment *from o to o' if and only if:*

$$\forall e \in Q_L(o), \exists e' \in Q_{L'}(o'); \langle e, e', = \rangle \in A$$

and, it is called an injective alignment *from o to o' if and only if:*

$$\forall e' \in Q_{L'}(o'), \exists e_1, e_2 \in Q_L(o); \langle e_1, e', = \rangle \in A \wedge \langle e_2, e', = \rangle \in A \implies e_1 = e_2$$

These properties heavily depend on the ontology entity languages which are chosen for alignments.

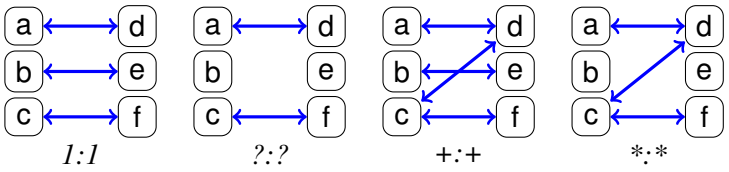
Usual mathematical properties apply to these alignments. In particular, a total alignment from o to o' is a surjective alignment from o' to o. A total alignment from both o and o', which is injective from one of them, is a bijection. In mathematical English, an injective function is said to be *one-to-one* and a surjective function to be *onto*. Due to the wide use among matching practitioners of the term *one-to-one* for a bijective, i.e., both injective and surjective, alignment, we will only use one-to-one for bijective.

Finally, we can extend these definitions when correspondence relations are *not* equivalence. In such a case, they do not ensure the same properties. For instance, injectivity does not guarantee reversibility of the alignment used as a transformation.

In conceptual models and databases, the term multiplicity denotes the constraints on a relation. Usual notations are 1:1 (one-to-one), 1:m (one-to-many), n:1 (many-to-one) or n:m (many-to-many). If we consider only total and injective properties, denoted as 1 for injective and total, ? for injective, + for total and * for none, and the two possible orientations of the alignments, from o to o' and from o' to o, the multiplicities become: ?:?, ?:1, 1:?, 1:1, ?:+, +:?, 1:+, +:1, +:+, ?:*, *:?, 1:*, *:1, +:*, *:+, *:* [EUZENAT and SHVAIKO 2013].

Example 35 (Alignment multiplicity)**.** *The alignment of Table 34 is ?:?. If we add the correspondence* Product $\geq$ Volume*, then it is ?:*. If we now consider relating any entity of the second ontology to another entity of the first one, then it becomes ?:+.*

The four pictures below display some of the possible configurations for two ontologies composed of three classes each.



8.3 Networks of ontologies

These definitions can be generalised to an arbitrary number of alignments and ontologies captured in the concept of a network of ontologies (or distributed system [GHIDINI and SERAFINI 1998; FRANCONI et al. 2003]), i.e., sets of ontologies and alignments.

Definition 63 (Network of ontologies)**.** *A network of ontologies $\langle \Omega, \Lambda \rangle$ is made of a set Ω of ontologies and a set Λ of alignments between these ontologies. We denote by $\Lambda(o, o')$ the set of alignments in Λ between o and o'.*

Example 36 (Network of ontologies)**.** *Figure 8.3 presents three ontologies (in all examples, $c \sqsubseteq c'$ denotes subsumption between concepts c and c', $c \bot c'$ denotes disjointness between concepts c and c', and $i \sqsubseteq c$ denotes membership of individual i to concept c):*

$$o_1 = \left\{ \begin{array}{l} b_1 \sqsubseteq a_1, c_1 \sqsubseteq a_1 \\ d_1 \sqsubseteq c_1, e_1 \sqsubseteq c_1 \end{array} \right\}$$

$$o_2 = \left\{ \begin{array}{ll} b_2 \sqsubseteq a_2, c_2 \sqsubseteq a_2, & g_2 \sqsubseteq b_2 \\ d_2 \sqsubseteq c_2, e_2 \sqsubseteq c_2, & f_2 \sqsubseteq b_2 \end{array} \right\}$$

$$o_3 = \left\{ \begin{array}{ll} b_3 \sqsubseteq a_3, c_3 \sqsubseteq a_3, & g_3 \sqsubseteq b_3 \\ d_3 \sqsubseteq c_3, e_3 \sqsubseteq c_3, & f_3 \sqsubseteq b_3 \\ i \sqsubseteq e_3, b_3 \bot c_3 & \end{array} \right\}$$

together with three alignments $A_{1,2}$, $A_{2,3}$, and $A_{1,3}$. These alignments can be described as follows:

$$A_{1,2} = \left\{ \begin{array}{l} b_1 \leq d_2 \end{array} \right\}$$

$$A_{2,3} = \left\{ \begin{array}{l} c_2 \leq b_3 \end{array} \right\}$$

$$A_{1,3} = \left\{ \begin{array}{l} e_1 \geq f_3, b_1 \geq e_3 \end{array} \right\}$$

The semantic web itself can be seen as a network of ontologies. Our goal is to make sense of these, i.e., to give them a semantics.

Hereafter, we consider normalised networks of ontologies, i.e., networks with exactly one alignment between each pair of ontologies.

Definition 64 (Normalised network of ontologies)**.** *A network of ontologies $\langle \Omega, \Lambda \rangle$ is said normalised if and only if for any two ontologies o and o', $|\Lambda(o, o')| = 1$.*

In a normalised network of ontologies, we denote by $\lambda(o, o')$ the unique alignment between o and o'.

Any network of ontologies may easily be normalised by:

- if $|\Lambda(o, o')| = 0$, adding an empty alignment between o and o',
- if $|\Lambda(o, o')| > 1$, replacing $\Lambda(o, o')$ by a unique alignment containing all the correspondences of the alignments of $\Lambda(o, o')$,

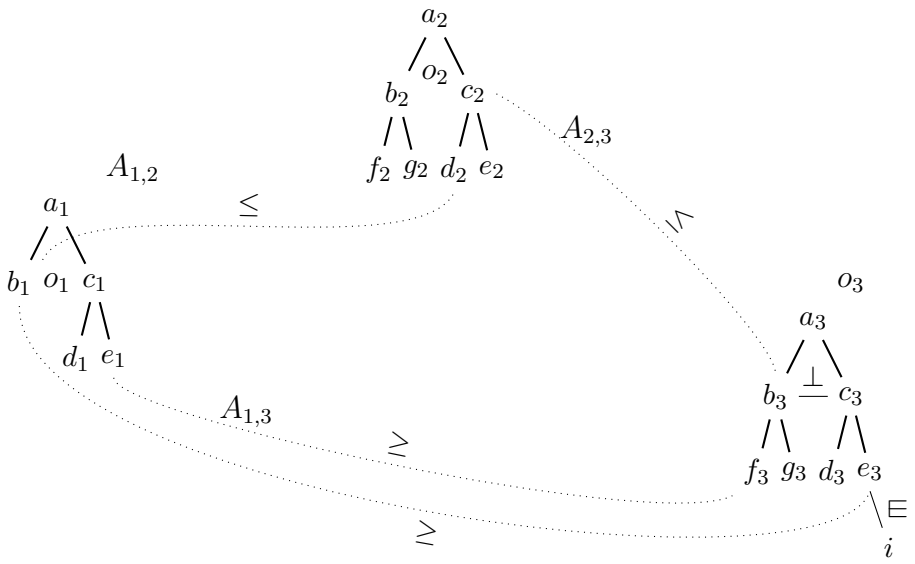


Figure 8.3: A network of ontologies made of three ontologies (o_1, o_2, and o_3) and three alignments ($A_{1,2}$, $A_{1,3}$, and $A_{2,3}$).

We call this standard normalisation:

Definition 65 (Standard normalisation)**.** *Given a network ontology $\langle \Omega, \Lambda \rangle$, its standard normalisation $\langle \Omega, \overline{\Lambda} \rangle$ is defined by $\forall o, o' \in \Omega$,*

$$\overline{\Lambda}(o, o') = \begin{cases} \{\{\}\}, & \text{if } \Lambda(o, o') = \varnothing, \\ \{\bigcup_{A \in \Lambda(o,o')} A\}, & \text{otherwise} \end{cases}$$

The unique element of $\overline{\Lambda}(o, o')$ is denoted by $\overline{\lambda}(o, o')$.

There are other ways to normalise such networks, but this simple one is sufficient for obtaining equivalent normalised networks (see Property 24).

Comparing networks of ontologies is not in general simple. For that purpose, we introduce the notion of morphism between two networks of ontologies.

Definition 66 (Syntactic morphism between networks of ontologies)**.** *Given two networks of ontologies, $\langle \Omega, \Lambda \rangle$ and $\langle \Omega', \Lambda' \rangle$, a* syntactic morphism *between $\langle \Omega, \Lambda \rangle$ and $\langle \Omega', \Lambda' \rangle$, is $\langle h, k \rangle$, a pair of morphisms: $h : \Omega \longrightarrow \Omega'$ and $k : \Lambda \longrightarrow \Lambda'$ such that $\forall o \in \Omega$, $\exists h(o) \in \Omega'$ and $o \subseteq h(o)$ and $\forall A \in \Lambda(o, o')$, $\exists k(A) \in \Lambda'(h(o), h(o'))$ and $A \subseteq k(A)$.*

Such a morphism exists when one network can be projected into another one an keep its structure and syntactic content. This means that any ontology (respectively any alignment) of the former has a counterpart in the latter one which contains at least all of its axioms (respectively correspondences) and that the graph structure of the former network is preserved in the latter. It is possible that several ontologies or alignments have the same counterpart as long as these conditions are met.

Morphisms can be used for defining syntactic subsumption between networks of ontologies.

Definition 67 (Syntactic subsumption between networks of ontologies)**.** *Given two networks of ontologies, $\langle \Omega, \Lambda \rangle$ and $\langle \Omega', \Lambda' \rangle$, $\langle \Omega, \Lambda \rangle$ is* syntactically subsumed *by $\langle \Omega', \Lambda' \rangle$, denoted by $\langle \Omega, \Lambda \rangle \sqsubseteq \langle \Omega', \Lambda' \rangle$, iff there exist a syntactic morphism between $\langle \Omega, \Lambda \rangle$ and $\langle \Omega', \Lambda' \rangle$.*

We note:

$$\langle \Omega, \Lambda \rangle \equiv \langle \Omega', \Lambda' \rangle \text{ iff } \langle \Omega, \Lambda \rangle \sqsubseteq \langle \Omega', \Lambda' \rangle \text{ and } \langle \Omega', \Lambda' \rangle \sqsubseteq \langle \Omega, \Lambda \rangle$$
$$\langle \Omega, \Lambda \rangle \sqsubset \langle \Omega', \Lambda' \rangle \text{ iff } \langle \Omega, \Lambda \rangle \sqsubseteq \langle \Omega', \Lambda' \rangle \text{ and } \langle \Omega', \Lambda' \rangle \not\sqsubseteq \langle \Omega, \Lambda \rangle$$

This definition is purely syntactic because semantically equivalent networks may not be syntactically subsumed (it suffices to use one ontology whose axioms are equivalent but different).

The empty network of ontologies $\langle \varnothing, \varnothing \rangle$ (containing no ontology and no alignment) is subsumed by any other network of ontologies.

It is possible to simplify the above definition in case of normalised networks of ontologies.

Property 21. *Given two normalised networks of ontologies $\langle \Omega, \Lambda \rangle$ and $\langle \Omega', \Lambda' \rangle$, $\langle \Omega, \Lambda \rangle \sqsubseteq \langle \Omega', \Lambda' \rangle$ iff $\exists h : \Omega \to \Omega'$ a morphism such that $\forall o \in \Omega$, $\exists h(o) \in \Omega'$ and $o \subseteq h(o)$ and $\forall o, o' \in \Omega$, $\lambda(o, o') \subseteq \lambda'(h(o), h(o'))$.*

Proof. $\Rightarrow$) In normalised networks, there always exists a single alignment between each pair of ontologies. Hence, if k is such that $\forall A \in \Lambda(o, o')$, $k(A) \in \Lambda'(h(o), h(o'))$, this means that $k(A) = k(\lambda(o, o')) = \lambda'(h(o), h(o'))$ and since $A \subseteq k(A)$, then $\lambda(o, o') \subseteq \lambda'(h(o), h(o'))$.

$\Leftarrow$) In $\langle \Omega, \Lambda \rangle$, $\Lambda(o, o') = \{\lambda(o, o')\}$ and the same holds for $\langle \Omega', \Lambda' \rangle$, if $k(\lambda(o, o')) = \lambda'(h(o), h(o'))$ it satisfies the constraint that $k(A) \in \Lambda'(h(o), h(o'))$ and $A \subseteq k(A)$. □

Moreover, networks are subsumed by their standard normalisation.

Property 22. *Let $\langle \Omega, \Lambda \rangle$ a network of ontologies and $\langle \Omega, \overline{\Lambda} \rangle$ its standard normalisation,*

$$\langle \Omega, \Lambda \rangle \sqsubseteq \langle \Omega, \overline{\Lambda} \rangle$$

Proof. Consider, the pair of morphisms $\langle h, k \rangle$ such that $h(o) = o$ and $\forall o, o' \in \Omega, \forall A \in \Lambda(o, o')$, $k(A) = \overline{\lambda}(o, o')$, then $A \subseteq k(A)$ because $A \subseteq \bigcup_{A' \in \Lambda(o,o')} A'$. □

From subsumption, conjunction (meet) can be introduced in a standard way:

Definition 68 (Syntactic conjunction of networks of ontologies)**.** *Given a finite family of networks of ontologies, $\{\langle \Omega_i, \Lambda_i \rangle\}_{i \in I}$, $\bigsqcap_{i \in I} \langle \Omega_i, \Lambda_i \rangle = \langle \Omega', \Lambda' \rangle$ such that $\forall i \in I$, $\langle \Omega', \Lambda' \rangle \sqsubseteq \langle \Omega_i, \Lambda_i \rangle$ and $\forall \langle \Omega'', \Lambda'' \rangle$; $\langle \Omega'', \Lambda'' \rangle \sqsubseteq \langle \Omega_i, \Lambda_i \rangle$, $\langle \Omega'', \Lambda'' \rangle \sqsubseteq \langle \Omega', \Lambda' \rangle$.*

Such a conjunction always exists because the empty network of ontologies is subsumed by all network of ontologies. However, it is likely not unique because of the choice of homomorphisms.

We can also define simple operations on networks of ontologies.

Definition 69 (Substitution in networks of ontologies)**.** *Given a network of ontologies $\langle \Omega, \Lambda \rangle$, given $o \in \Omega$ and o' another ontology, $\langle \Omega, \Lambda \rangle[o/o'] = \langle \Omega \setminus \{o\} \cup \{o'\}, \Lambda \setminus \bigcup_{o'' \in \Omega} (\Lambda(o, o'') \cup \Lambda(o'', o)) \rangle$.*

Given a network of ontologies $\langle \Omega, \Lambda \rangle$, given $A \in \Lambda(o, o')$ and A' another alignment between o and o', $\langle \Omega, \Lambda \rangle[A/A'] = \langle \Omega, \Lambda \setminus \{A\} \cup \{A'\} \rangle$.

Property 23. *Given a network of ontologies $\langle \Omega, \Lambda \rangle$ with $o \in \Omega$ and $A \in \Lambda$, if $o' \subseteq o$, then $\langle \Omega, \Lambda \rangle[o/o'] \sqsubseteq \langle \Omega, \Lambda \rangle$ and if $A' \subseteq A$, then $\langle \Omega, \Lambda \rangle[A/A'] \sqsubseteq \langle \Omega, \Lambda \rangle$*

Proof. Simply, there exists a pair of morphisms $\langle h, k \rangle$ which identifies each ontology to itself but o' which is identified to o, and each alignment to itself, but A' which is identified to A in the second case. In the case of $\langle \Omega, \Lambda \rangle[o/o']$, the set of alignments is strictly included in Λ. It is thus clear that $h(o'') \subseteq o''$ and $k(A'') \subseteq A''$, including for o' and A' (by hypothesis). The structure is preserved because ontologies and alignments are the same, except in the second case in which the substituted alignment preserves the structure. □

8.4 Conclusion

We presented a simple syntax for expressing alignments between ontologies and networks of alignments. More complex syntaxes may be designed if necessary, for instance for dealing with complex entity languages when the ontology languages are not very expressive.

We now extend this syntactic part by presenting a convenient way to deal with relations between entities in alignments.

Chapter 9

Alignment algebra

The next important component of the alignment is the relation that holds between the entities. We identify a set of relations Θ that is used for expressing the relations between the entities. Matching algorithms primarily use the equivalence relation ($=$) meaning that the matched objects are the same or are equivalent if these are formulas. It is possible to use relations from the ontology language within Θ. For instance, using OWL, it is possible to take advantage of the `owl:equivalentClass`, `owl:disjointWith` or `rdfs:subClassOf` relations in order to relate classes of two ontologies. These relations correspond to set-theoretic relations between classes: *equivalence* ($=$), *disjointness* ($\perp$), *less general* ($\sqsubseteq$). They can be used without reference to any ontology language.

Example 37 (Alignment)**.** *Consider two alignments A_1 and A_2, relating respectively the German to the French ontology and the French ontology to the British one, containing the following correspondences (A_1 is on the left, A_2 on the right):*

$$\textit{Konstruktion} \perp \textit{Commune} \qquad \textit{Commune} \geq \textit{Municipality}$$
$$\textit{Stadtgebiet} > \textit{Ville} \qquad \textit{Ville} \between \textit{Municipality}$$

This means that A_1 considers that a Konstruction, i.e., a Building, is disjoint from a Commune, i.e., a Ward, and a Stadtgebiet, i.e., a Urban area, is more general than a Ville, i.e., a Town. A_2 expresses that a Commune is more general or equivalent to a Municipality and Ville overlaps withMunicipality, i.e., that both concepts have common instances but none is more general than the other.

This definition does not tell how to interpret this set of correspondences. However, it is clear from usage that it has to be interpreted in a conjunctive manner: all the correspondences are asserted to hold when asserting an alignment.

Hence, the problem of expressing disjunctions of correspondences can be raised. This can be because it is necessary to aggregate the result of methods which address the ontology matching problem from different dimensions, this can be because the person or the program generating the alignment is unsure about the exact relation but knows that this relation is constrained to a specific set of alternative relations.

Example 38 (Disjunctive relations)**.** *For instance, an engineer may know that a Stadt, i.e., Town, and a Town are similar things but may not know exactly the nature of the overlaps. She can express that they are not disjoint by the disjunction of relations $<$, $>$, $\between$ and $=$, thus prohibiting $\perp$. This can also be because the alignment has been generated by composing two alignments. This operation does not usually return a simple relation but a disjunction of such relations, e.g., if Stadtgebiet, i.e., Urban area, is more general than Ville, i.e., Town, and Ville overlaps Municipality, then Stadtgebiet either is more general or overlaps Municipality, it cannot be disjoint with it. Hence, the result is a disjunction of relations.*

This is also the case of the $\geq$ (more-general-or-equal) and $\leq$ (more-specific-or-equal) relations used by some systems. These are typically the disjunction of $<$ and $=$ or $>$ and $=$. In fact, practice which considers that if both $\leq$ and $\geq$ hold (conjunction), then $=$ holds, only reflects the set operation: $\{<,=\} \cap \{>,=\} = \{=\}$ or the logical interpretation that:

$$\forall a, b, (a < b \vee a = b) \wedge (a > b \vee a = b) \models a = b \text{ if } <, > \text{ and } = \text{ are exclusive}$$

In order to apply a systematic treatment for disjunctive alignment relations, we use algebra of relations and we show that this has many advantages.

9.1 Algebras of relations

An algebra of binary relations (hereafter referred to as relation algebra[1]) [TARSKI 1941] is a structure $\langle \Theta, \wedge, \vee, \neg, \top, \bot, *, \breve{}, 1' \rangle$ such that $\langle \Theta, \wedge, \vee, \neg, \top, \bot \rangle$ is a Boolean algebra; $*$ is an associative internal composition law with (left and right) unity element $1'$, that distributes over $\vee$; $\breve{}$ is an internal involutive unary operator, that distributes over $\vee$, $\wedge$ and *.

We consider a particular type of relation algebras[2] in which Θ is the powerset of a generating set Γ closed under $\neg$ and $\wedge / \vee / \neg$ are set intersection/union/complementation ($\cap / \cup / \Gamma -$). Such an algebra of (binary) relations is defined by $\langle 2^\Gamma, \cap, \cup, \Gamma -, \Gamma, \varnothing, \cdot, ^{-1}, \{=\} \rangle$ such that:

- Γ is a set of jointly exhaustive and pairwise disjoint (JEPD) relations between two entities. This means that, in any situation, the actual relation between two objects is one and only one of these relations. Sets of relations allow to express uncertainty: the full Γ set is the "I do not know" relation since it is satisfied by any pair of entities;
- $\cap$ and $\cup$ are set operations used to meet and join two sets of base relations, hence if xry or $xr'y$, then $xr \cup r'y$;
- $\cdot$ is the composition operator such that if xry and $y\ r'\ z$, then $x\ r \cdot r'\ z$; "=" is such that $\forall r \in \Gamma, (r \cdot =) = (= \cdot r) = r$;
- $^{-1}$ is the converse operator, i.e., such that $\forall e, e' \in \Gamma, ere' \Leftrightarrow e'r^{-1}e$.

These operations are applied to sets of base relations by distributing them on each element, e.g., $R \cdot R' = \bigcup_{r \in R, r' \in R'} r \cdot r'$.

A typical example of such an algebra is the Allen algebra of temporal interval relations [ALLEN 1983]. Here, we will consider, as an example, a simpler algebra, called $A5$, isomorphic to that applying to sets in which the base (JEPD) relations between two sets are equivalent ($=$), includes ($>$), is-included-in ($<$), overlaps ($\between$) and disjoint ($\bot$). In this algebra, all base relations but $<$ and $>$ are their own converse while $>^{-1}=<$ and $<^{-1}=>$.

The complete set of $2^5 - 1 = 31$ valid relations that can be made out of these 5 base relations is depicted in Figure 9.1. Among these relations, Γ means "I do not know" as it contains all the base relations. $\neq$ is equivalent to $\{<, >, \between, \bot\}$, $\leq$ is equivalent to $\{=, <\}$, $\geq$ to $\{=, >\}$, $\not\approx$ to $\{<, >, \between\}$ and $\not\perp$ to $\{=, <, >, \between\}$. The composition table is given in Table 9.1.

Relation algebras can still be used when the ontology entities are formulas (or queries) and the base relations are logical connectives between formulas ($\Rightarrow$, $\equiv$). Indeed, it is sufficient to split the disjunctive relations into a disjunction of formulas:

$$\phi\{\Rightarrow, \equiv\}\psi \text{ would be equivalent to } \phi \equiv \psi \vee \phi \Rightarrow \psi$$

[1] There was a mistake in the presentation of [EUZENAT 2008] which merged negation and inverse.

[2] [LIGOZAT and RENZ 2004] shows that a weaker structure than algebras of relations, non associative, can be used in most of the purposes of qualitative calculi. However, we will need associativity later.

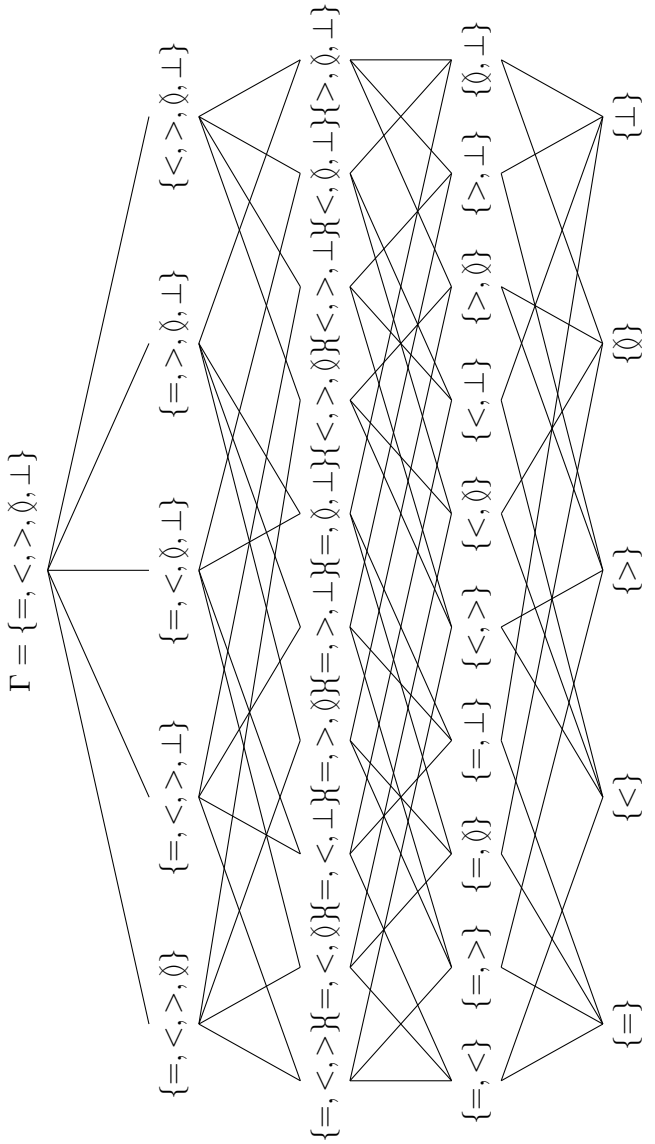


Figure 9.1: The lattice of 31 disjunctive relations in $A5$.

(these relations are not exclusive anymore).

9.2 Aggregating matcher results

The set Θ is closed for $\vee$, $\wedge$ and $\neg$. This means that any combination of these operations yields an element of Θ. This is very powerful if one wants to combine relations, e.g., for combining correspondences and alignments.

When matching methods bring new evidences for a correspondence from a different perspective, they are though of as bringing new arguments in favor of a correspondence. Hence, its results must be aggregated with union ($R \cup R'$). When the matching methods, instead, are competing algorithms providing all the possible base relations, i.e., providing arguments against the non selected correspondences, then intersection ($R \cap R'$) should be used. Because, we now have two distinguished operations, they can be used together in the same application.

These operations can be used for describing two cases of matching process: an expanding matching process which starts with the empty relation ($\varnothing$) between each pair of entities and which finds evidences for more base relations between these entities aggregating them with $\cup$ and a contracting process which starts with Γ between each pair of entities and which discards support for some base relations between entities aggregating the result with $\cap$. In the first case, the more matching methods are used, the less precise the alignment becomes: this can be balanced by confidence measures as we will see below. In the second case, the more methods are used, the more precise the alignment becomes.

These operations on Θ are used in correspondence aggregation: an alignment is interpreted as all the correspondences it contains hold, and a distributed system is interpreted as all alignments hold. Hence, the disjunctive aggregation of alignments is based on the combination of their set of correspondences with the union of relations; the conjunctive aggregation of alignments is based on the combination of their set of correspondences with the intersection of relations. Hence, we define a normalisation operation $\bar{A}$, which implements the conjunctive interpretation of alignments. It provides exactly one correspondence per pair of entities and makes explicit all the relations between entities in A:

$$A^0 = \{\langle e, e', \Gamma\rangle | e \in Q_L(o), e' \in Q_{L'}(o')\}$$
$$\bar{A} = \{\langle e, e', \cap_{\langle e,e',r\rangle \in A \cup A^0} r\rangle\}$$

It is then easy to define intersection:

$$A \wedge A' = \{\langle e, e', r \cap r'\rangle | \langle e, e', r\rangle \in \bar{A}, \langle e, e', r'\rangle \in \bar{A'}\}$$

as well as additional operators such as disjunction and converse of alignments:

$$A \vee A' = \{\langle e, e', r \cup r'\rangle | \langle e, e', r\rangle \in \bar{A}, \langle e, e', r'\rangle \in \bar{A'}\}$$
$$A^{-1} = \{\langle e', e, r^{-1}\rangle | \langle e, e', r\rangle \in \bar{A}\}$$

Below, alignments are always presented in a reduced normalised way, i.e., without trivial $\langle e, e', \Gamma\rangle$ correspondences added by normalisation.

Example 39 (Alignment aggregation)**.** *Consider two alignments A_3 and A_5, resulting from two different matchers which match ontologies using different features for ruling out correspondences (A_3 is on the left, A_5 on the right):*

Konstruktion$\{\bot\}$*Municipality*	*Stadt*$\{<\}$*Town*
Stadtgebiet$\{>, \emptyset\}$*Municipality*	*Stadtgebiet*$\{\bot, \emptyset\}$*Municipality*

	$=$	$<$	$>$	$\between$	$\bot$
$=$	$=$	$<$	$>$	$\between$	$\bot$
$<$	$<$	$<$	$=<>\between$	$>\between\bot$	$<\between\bot$
$>$	$>$	Γ	$>$	$<\between$	$\bot$
$\between$	$\between$	$<\between\bot$	$>\between$	Γ	$<\between\bot$
$\bot$	$\bot$	$\bot$	$>\between\bot$	$>\between\bot$	Γ

Table 9.1: Composition table for the A_5 relation algebra (read first relation determining the column and second relation determining the line: the table is inverted with respect to [EUZENAT 2008]).

Since these matchers provide competing alignments between ontology o and o', their result can be aggregated conjunctively. The result $A_6 = A_3 \wedge A_5$, is given below as the left-hand side alignment:

$$\begin{array}{rcl} \textit{Konstruktion}\{\bot\}\textit{Municipality} & & \\ \textit{Stadt}\{<\}\textit{Town} & & \textit{Stadt}\{=,<,>,\between\}\textit{Town} \\ \textit{Stadtgebiet}\{\between\}\textit{Municipality} & & \textit{Stadtgebiet}\{\bot\}\textit{Municipality} \end{array}$$

This alignment is aggregated with the right-hand side alignment A_4. Since they provide evidence for alignments from different perspective, they are aggregated disjunctively, yielding $A_8 = A_4 \vee A_6$:

$$\begin{array}{c} \textit{Stadt}\{=,<,>,\between\}\textit{Town} \\ \textit{Stadtgebiet}\{\between,\bot\}\textit{Municipality} \end{array}$$

Algebras of relations are useful because they can account for these two behaviours. However, there are other benefits brought by algebra of relations.

9.3 Composing alignments

Another way of reusing alignments is to deduce new alignments from existing ones. One way to do so, is to compose alignments. If there exists an alignment between ontology o and ontology o'', and another alignment between o'' and a third ontology o', we would like to find which correspondences hold between o and o'. The operation that returns this set of correspondences is called composition.

Alignment composition has already considered [ZIMMERMANN, KRÖTZSCH, et al. 2006] with the idea that, in an open system like the Alignment API, the rules for composing alignment relations, e.g., $\mathsf{instanceOf} \cdot \mathsf{subClassOf} = \mathsf{instanceOf}$, should be given by a composition table. Composition tables come directly from algebra of relations and they naturaly extend from base relations to disjunctions of base relations.

Alignment composition can thus be reduced to combining correspondences with regard to their relations and the structure of related entities and computing the confidence degree of the result. The composition table between the base relations of $A5$ is given in Table 9.1.

The composition of two alignments A and A' is defined by:

$$A \cdot A' = \overline{\{\langle e, e'', r \cdot r' \rangle | \langle e, e', r \rangle \in A, \langle e', e'', r' \rangle \in A'\}}$$

One can compose an alignment with itself (self-composition) through: $A^2 = A \cdot A^{-1} \cdot A$. This operation may provide new correspondences.

Example 40 (Composing alignments)**.** *The alignment A_3 of Example 39, is the result of the composition of alignments A_1 and A_2 of Example 37: $A_3 = A_1 \cdot A_2$. The first simple application of Table 9.1 occurs*

when composing Konstruktion $\{\perp\}$ Commune *and* Commune $\{>,=\}$ Municipality*, then it can be deduced that* Konstruktion $\{\perp\}$ Municipality *because* $\{\perp\}\cdot\{>,=\} = (\perp\cdot >)\cup(\perp\cdot =) = \{\perp\}$*. Things can be more complex, when composing* Stadtgebiet $\{>\}$ Ville *and* Ville $\{\lozenge\}$ Municipality*, then Table 9.1 allows to deduce that* Stadtgebiet $\{>,\lozenge\}$ Municipality *because* $\{>\}\cdot\{\lozenge\} => \cdot\ \lozenge = \{<,\lozenge\}$*. The result provided by the table in this case is a disjunction of relations because it is not possible to obtain more precise information from the alignments alone.*

Very often the composition of two base relations is not a base relation but a disjunction of relations. Hence, if we were not dealing with sets of base relations, it would not be possible to represent the composition of two alignments by an alignment.

Moreover, defining composition by an algebra of relations automatically satisfies all the constraints on the categorical characterisation of alignments defined in [ZIMMERMANN, KRÖTZSCH, et al. 2006]: it must be associative and have an identity element. This is true from the definition of algebra of relations.

9.4 Algebraic reasoning with alignments

α-consequences are correspondences which are entailed by two aligned ontologies [EUZENAT 2007]; they can be extended as the correspondences entailed by a system of many ontologies and many alignments between them. [ZIMMERMANN and LE DUC 2008] introduced the notion of quasi-consequences as the set of formulas entailed by a set of alignments alone (without considering ontologies). This notion can be straightforwardly extended to correspondences as quasi-α-consequences: the correspondences which are entailed by the set of alignments, when considering the ontologies as void of axioms. Quasi-α-consequences are also α-consequences.

Reasoning on alignments aims at using existing alignments in order to deduce more and more complete alignments. Such a reasoning procedure can be considered, for soundness and completeness, with respect to α-consequences.

Algebraic reasoning (using combination of composition, converse, and intersection) can be used as a practical and efficient way to reason with alignments. The algebraic closure of a set of alignments S is the set of normalised alignments, containing $\bar{S}$, closed under composition, converse and intersection.

This procedure is correct (the algebraic operations can be transformed into their logical equivalent). However, since it does not consider ontologies, it can only deduce quasi-α-consequences. We have no guarantee that it is complete even for finding quasi-α-consequences.

However, this can already be used for two purposes: (1) improving the existing alignments by deducing new correspondences coming either from the alignment itself or from other alignments, and (2) checking the consistency of a set of alignments (or one alignment). Indeed, if we can deduce $x\{\}y$, e.g., because $x\{<\}y$ and $x\{>\}y$ for two competing matchers, since the intersection is empty we know that the alignment itself is inconsistent. This kind of reasoning can be more complex, involving several alignments as well as composition operations, i.e., checking a whole distributed system. Then, the set of alignments as a whole would be inconsistent, hence the distributed system would have no model.

Example 41 (Algebraic reasoning)**.** *The simplest instance of an inconsistent alignment is to have two contradictory statements like* Konstruktion$\{\perp\}$Town *and* Konstruktion$\{<\}$Town *in the same alignment. The conjunction of these two relations, obtained by normalisation, is empty. Such an inconsistent alignment can also be obtained by combining consistent alignments. For instance, aggregating conjunctively alignments A_3 and A_4 of Example 39, will generate the inconsistent* Stadtgebiet $\{\}$ Municipality *correspondence.*

Algebraic reasoning allows to expand alignments. For instance, Stadt$\{>\}$Town*,* Stadtgebiet$\{<\}$Town*,* Stadtgebiet$\{\perp\}$Municipality *entails* Stadt$\{>,\lozenge,\perp\}$Municipality*. Computing the compositional closure of this alignment will find this correspondence. Moreover, if the initial alignment also contain* Stadt$\{=,<\}$Municipality*, the compositional closure will bring the inconsistence to light.*

Once again, it is possible to use disjunctive relations to better evaluate alignments. Indeed, the problem is that if a matcher returns a correspondence between two entities with relation $\leq$ while the expected (and exact) relation was $<$, then the use of syntactic precision and recall measures would count this relation as incorrect. Hence, if the expected alignment was made of this correspondence alone, both precision and recall would be 0. This is unfair because it cannot be said that this correspondence is both incorrect and incomplete. In fact, it is incomplete, because it does not provide the exact relation, but not incorrect, because the relation is more general than the correct one.

This is indeed what happens with semantic precision and recall [EUZENAT 2007]: since the relation $\{<, =\}$, which is the disjunction of $<$ and $=$, can be deduced from $\{<\}$ alone, it would count as correct for semantic precision and still as incorrect for semantic recall because $\{<, =\}$ does not entail $\{<\}$.

We could introduce an algebraic precision and recall for evaluating ontology alignments as an intermediary step between classic precision and recall and semantic precision and recall. It would simply use the inclusion between the relations as suggested above instead of the entailment between correspondences of [EUZENAT 2007] and would be far easier to compute. The resulting measure would be a relaxation of precision and recall in the sense of [EHRIG and EUZENAT 2005].

9.5 Algebra granularity

In order to investigate granularity within algebras of relations, we introduced the notion of weakening [EUZENAT 2001]. Weakening an algebra of relations simply consists of grouping together several base relations and taking the result as the base relations of the, less precise, weaker algebra.

In fact, taking any maximal antichain[3] that preserves converse in the lattice of Figure 9.1 yields a base for an algebra of relations. Other constraints can be put on weakening, such as requiring that they preserve a neighbourhood structure. Neighbourhood structures for algebras of relations have been introduced in [FREKSA 1992]. They are based on a connectivity relation between relations that is used for defining neighbourhood. This connectivity relation can be based on different properties of the domain the relations apply to. We have shown that granularity operators, at least in time and space algebras, can be automatically built on such neighbourhoods [EUZENAT 2001].

In terms of alignment, the interesting aspect of this weakening operation is that it helps considering that alignments using different sets of relations are compatible and can still be used together. Coming back to the example of set-relations, there can be systems that provides only the $=$ base relation leaving implicitly all the others as Γ, there are other systems like the one considered previously which consider $=$, $<$, $>$, $\between$ and $\perp$. The set of base relations is different and thus it is not easy to combine two such alignments. However, if we consider that the first one is a weakening of the second one (grouping $<$, $>$, $\between$ and $\perp$ into $\neq$), then it is possible to import one alignment into another formalism and vice-versa (at the expense of completeness when we export to the weaker algebra).

Example 42 (Algebra granularity)**.** *Different sources of alignments may provide alignments with different kinds of relations between objects. For instance, the following A_7 alignment (left-hand side) is expressed in the simple $\{\perp, \not\perp\}$ algebra, i.e., identifying only incompatible elements.*

Stadt$\{\not\perp\}$*Town*	*Stadt*$\{\not\perp\}$*Town*
Stadtgebiet$\{\perp\}$*Municipality*	*Stadtgebiet*$\{\perp, \not\perp\}$*Municipality*

Thanks to the use of compatible algebras, A_7 can be expressed in the more expressive algebra. In fact, the alignment A_4 of Example 39 is the transcription of A_7 in the $A5$ algebra. On the other hand, it is possible to degrade an alignment into the coarser algebra at the expense of precision. The alignment on the right-hand side is the result of converting the alignment A_5 of Example 39 to the $\{\perp, \not\perp\}$ algebra.

[3] An antichain is a set of relations such that no one is comparable to the other.

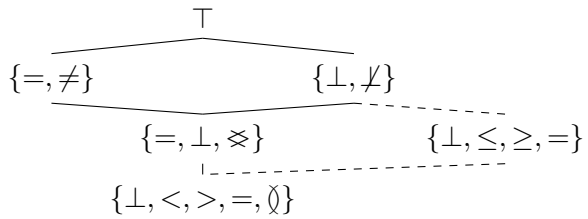


Figure 9.2: The reasonable weakenings of $A5$ ($\not= = \{<, >, =, \lozenge\}$ and $\not\not= = \{<, >, \lozenge\}$).

Figure 9.2 shows the "interesting" weakened algebras of relations from the initial algebra. It features the $\{=, \neq\}$ algebra but also shows that the usually considered $\{=, \leq, \geq, \perp\}$ is not a correct base for such an algebra because it is neither jointly exhaustive ($\leq$ and $=$ can occur at the same time), nor pairwise disjoint ($\lozenge$ is missing).

9.6 Conclusion

We briefly introduced algebras of alignment relations in order to deal with disjunction of base relations in alignments. They are very powerful for manipulating alignments and network of ontologies without accessing the ontology themselves.

The theory of such algebras remain to be completed, in particular in relation to confidence measures [ATENCIA, BORGIDA, et al. 2012; INANTS, ATENCIA, et al. 2016] and to combining algebras of heterogeneous relations [INANTS and EUZENAT 2015].

Ontology matching

Ontology matching is the task of finding an alignment between two ontologies [EUZENAT and SHVAIKO 2013]. It is depicted in the diagram of Figure 9.3.

Numerous algorithms have been designed to fulfil this task exploiting techniques developed in various domains. Ontology matching approaches may be organised in:

- content-based approaches which rely on ontology content. These may be based on:
  - Labels used to name or comment the entities found in the ontologies. Techniques may be inspired from computational linguistics or information retrieval.
  - Relationships between entities, and prominently the subsumption relation. Techniques from graph theory may be used.
  - Concept extensions, i.e., their instances when available. In this case, machine learning, data analysis or deduplication techniques are used.
  - Semantics of ontologies which may be used for expanding alignments or testing their consistency. Then, techniques from automated reasoning and constraint programming may be used.
- context-based approaches which rely on relationships that the ontologies have with other resources. These may be:
  - a corpus of documents annotated by one or both of the ontologies which allows for exploiting statistical learning;
  - one or several ontologies used as background knowledge for deducing relations between entities through automated reasoning;
  - user feedback when these ontologies are allowed to improve alignments;
  - the relation with specific resources such as DBPedia, WordNet or UMLS.

However, none of these approaches works satisfactorily in all situations. Hence, systems use several such techniques together. It is not possible to present these different types of systems (see [EUZENAT and SHVAIKO 2013]). We briefly consider below those using machine learning.

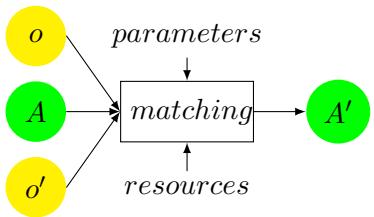


Figure 9.3: The ontology matching process: it establishes an alignment (A') from two ontologies (o and o') and optionally an input alignment (A), parameters and external resources (from [EUZENAT and SHVAIKO 2013]).

Ontology matching is an inductive rather then deductive task. There are no rule for deducing alignments, only heuristics to finding them. It is thus natural to reuse machine learning techniques to match ontologies. Two types of techniques may contribute: data mining, which isolates regularities in large quantity of data, and learning from example, which induces correspondence patterns from examples of correspondences.

Learning from examples is less frequent because, as mentioned, matching cannot easily be generalised. It can, however, be used in two cases: first, when a user is able to show some simple alignments from which the system can learn how to match, and, second, when there are references alignments from which it is possible to learn algorithm parameters which provide the best results [EHRIG, STAAB, et al. 2005].

Data mining algorithms infer the similarity between two concepts from the similarity between their instances, or the probability that the instances of one class be instances of another one. This may be based on symbolic techniques such as formal concept analysis [STUMME and MÄDCHE 2001; KALFOGLOU and SCHORLEMMER 2003], decision trees [XU and EMBLEY 2003] or numeric techniques such as neural networks [MAO et al. 2010], Bayes networks [PAN et al. 2005], association rule extraction [DAVID et al. 2007] or a mix of such approaches [DOAN et al. 2004].

Chapter 10

Alignment semantics

If alignments have to be used with ontologies, they have to be given a semantics. The problem is to provide a semantics to aligned ontologies or networks of ontologies. This semantics must play correctly with that of ontologies, because connecting ontologies to other ontologies should be compatible with the semantics of ontologies.

10.1 Reminder: ontology semantics

We consider ontologies as logical theories that may contain ground assertions. The languages used in the semantics web such as RDF or OWL are indeed logics [HITZLER et al. 2009; ANTONIOU, GROTH, et al. 2012]. An ontology is a set of assertions that selects the set of interpretations which satisfy them. These interpretations are called models. They constitute the possible interpretations of an ontology.

Definition 70 (Model)**.** *Given an ontology o, a model of o is an interpretation m of o, which satisfies all the assertions in o:*

$$\forall \delta \in o, m \models \delta$$

The set of models of an ontology o is denoted as $\mathcal{M}(o)$.

Given a model m, we will denote as $m(e)$ the application of the interpretation function of the model to some ontology entity e.

Finally, an important notion is the set of assertions that are consequences of an ontology. These are the assertions implicitly entailed by an ontology and they determine the answers to queries.

Definition 71 (Consequence)**.** *Given an ontology formula δ, δ is a consequence of an ontology o, if and only if, it is satisfied by all models of o. This is denoted as $o \models \delta$.*

Such a relation satisfies three properties (o, o' are ontologies, i.e., sets of assertions, δ and γ are assertions):

extensivity $\{\delta\} \models \delta$
monotony if $o \models \delta$ then $o \cup o' \models \delta$
idempotency if $o \models \delta$ and $o \cup \{\delta\} \models \gamma$ then $o \models \gamma$

We assume a consequence closure function $Cn^{\omega}(o) = \{\delta \mid o \models \delta\}$.

This digression introduced more precisely, albeit generally, a simplified syntax and semantics of ontologies. This will be useful when considering the meaning of matching ontologies.

10.2 General alignment semantics

Intuitively, an alignment expresses constraints between ontologies. It thus restricts the semantics of these ontologies to satisfy these constraints. The general semantic framework consider here will restrict the ontology models to those compatible with the alignments of the networks.

When ontologies are independent, i.e., not related with alignments, it is natural that their semantics is the classical semantics for these ontologies, i.e., a set of models $\mathcal{M}(o)$. A model is a map m from the entities of the ontologies to a particular domain D. Very often, this domain contains the powerset and the powerset of the product of a particular universe of discourse (classes are interpreted on the powerset and relations on the product). Such models have to apply to all the elements of the entity language $Q_L(o)$ (when it is larger than the ontology language, this is usually defined inductively on the structure of its elements).

Different semantics provide alternative ways to record the constraints imposed by alignments: through relations between domains of interpretation [GHIDINI and SERAFINI 1998; BORGIDA and SERAFINI 2003], through equalising functions [ZIMMERMANN and EUZENAT 2006; ZIMMERMANN 2008], by imposing equal [LENZERINI 2002] or disjoint [CUENCA GRAU et al. 2006] domains. These models have been compared in [ZIMMERMANN and EUZENAT 2006]. Here, we provide an informal unified view of these semantics (see Chapter 11 for description of one of these particular semantics).

For that purpose, each correspondence is interpreted with respect to three features: a model for each ontology and a semantic structure, denoted by Δ [ZIMMERMANN 2013]. This loosely defined semantic structure has two purposes:

- providing an interpretation to the correspondence relations in Θ (which are independent from the ontology semantics);
- memorising the constraints imposed on models by the alignments.

In this work, it can simply be considered that Δ is used, in each semantics, to define the satisfaction of a correspondence μ by two ontology models o and o' (which is denoted by $m_o, m_{o'} \models_\Delta \mu$).

Such a simple notion of satisfaction, imposing no constraints on models, is provided by Example 43.

Example 43 (Interpretation of correspondences)**.** *In the language used as example, c and c' stand for classes and i and i' for individuals. If m_o and $m_{o'}$ are respective models of o and o':*

$$
\begin{aligned}
m_o, m_{o'} &\models_\Delta \langle c, c', = \rangle \textit{ iff } m_o(c) = m_{o'}(c') \\
m_o, m_{o'} &\models_\Delta \langle c, c', \leq \rangle \textit{ iff } m_o(c) \subseteq m_{o'}(c') \\
m_o, m_{o'} &\models_\Delta \langle c, c', \geq \rangle \textit{ iff } m_o(c) \supseteq m_{o'}(c') \\
m_o, m_{o'} &\models_\Delta \langle i, c', \in \rangle \textit{ iff } m_o(i) \in m_{o'}(c') \\
m_o, m_{o'} &\models_\Delta \langle c, i', \ni \rangle \textit{ iff } m_{o'}(i') \in m_o(c) \\
m_o, m_{o'} &\models_\Delta \langle c, c', \bot \rangle \textit{ iff } m_o(c) \cap m_{o'}(c') = \varnothing
\end{aligned}
$$

Example 44 (Interpretation of correspondences in first-order logic)**.** *The semantics can be given with respect to first-order logic theories. In such a case, correspondences relate predicates p and p' of the same arity between ontologies o and o'. If m_o and $m_{o'}$ are their respective first-order models:*

$$
\begin{aligned}
m_o, m_{o'} &\models_\Delta \langle p, p', = \rangle \textit{ iff } m_o(p) = m_{o'}(p') \\
m_o, m_{o'} &\models_\Delta \langle p, p', \leq \rangle \textit{ iff } m_o(p) \subseteq m_{o'}(p') \\
m_o, m_{o'} &\models_\Delta \langle p, p', \geq \rangle \textit{ iff } m_o(p) \supseteq m_{o'}(p') \\
m_o, m_{o'} &\models_\Delta \langle p, p', \bot \rangle \textit{ iff } m_o(p) \cap m_{o'}(p') = \varnothing
\end{aligned}
$$

This semantics selects first-order theory interpretations by setting constraints on predicate interpretations.

Example 45 (Interpretation of correspondences in the equalising semantics)**.** *An alternative interpretation in the equalising semantics (see Chapter 11) relies on a family of functions indexed by each ontology γ from the domains of interpretations of each ontologies to a universal domain U. Hence, $\Delta = \langle \gamma, U \rangle$. Then, in the case of Example 43, if m_o and $m_{o'}$ are respective models of o and o′:*

$$m_o, m_{o'} \models_\Delta \langle c, c', = \rangle \textit{ iff } \gamma_o \circ m_o(c) = \gamma_{o'} \circ m_{o'}(c')$$
$$m_o, m_{o'} \models_\Delta \langle c, c', \leq \rangle \textit{ iff } \gamma_o \circ m_o(c) \subseteq \gamma_{o'} \circ m_{o'}(c') \textit{ or } \gamma_o \circ m_o(c) \in \gamma_{o'} \circ m_{o'}(c')$$
$$m_o, m_{o'} \models_\Delta \langle c, c', \geq \rangle \textit{ iff } \gamma_o \circ m_o(c) \supseteq \gamma_{o'} \circ m_{o'}(c') \textit{ or } \gamma_{o'} \circ m_{o'}(c') \in \gamma_o \circ m_o(c)$$
$$m_o, m_{o'} \models_\Delta \langle i, c', \in \rangle \textit{ iff } \gamma_o \circ m_o(i) \in \gamma_{o'} \circ m_{o'}(c') \textit{ or } \gamma_o \circ m_o(i) \subseteq \gamma_{o'} \circ m_{o'}(c')$$

This semantics allows for changing the interpretation of an individual as a set and vice-versa.

Hence, the semantics of two aligned ontologies may be given as a set of models which are pairs of compatible models.

Definition 72 (Models of alignments)**.** *Given two ontologies o and o′ and an alignment A between these ontologies, a model of this alignment is a triple $\langle m_o, m_{o'}, \Delta \rangle$ with $m_o \in \mathcal{M}(o)$, $m_{o'} \in \mathcal{M}(o')$, and Δ a semantic structure, such that $\forall \mu \in A$, $m_o, m_{o'} \models_\Delta \mu$ (denoted by $m_o, m_{o'} \models_\Delta A$).*

We note $A \models \mu$ iff $\forall \langle m_o, m_{o'}, \Delta \rangle$ such that $m_o, m_{o'} \models_\Delta A$, $m_o, m_{o'} \models_\Delta \mu$. Similarly as for ontologies, the semantics of alignments can be given by the relation $\models$ such that (A and A' are alignments and μ and ν are correspondences all between the same pair of ontologies):

extensivity $\{\mu\} \models \mu$
monotony if $A \models \mu$ then $A \cup A' \models \mu$
idempotency if $A \models \mu$ and $A \cup \{\mu\} \models \nu$ then $A \models \nu$

Similarly, we assume a consequence closure function $Cn^\alpha(A) = \{\mu \mid A \models \mu\}$.

Models of networks of ontologies extend models of alignments. They select compatible models for each ontology in the network [GHIDINI and GIUNCHIGLIA 2001]. Compatibility consists of satisfying all the alignments of the network.

Definition 73 (Models of networks of ontologies)**.** *Given a network of ontologies $\langle \Omega, \Lambda \rangle$, a model of $\langle \Omega, \Lambda \rangle$ is a pair $\langle m, \Delta \rangle$ with m a family of models indexed by Ω with $\forall o \in \Omega$, $m_o \in \mathcal{M}(o)$ such that for each alignment $A \in \Lambda(o, o')$, $m_o, m_{o'} \models_\Delta A$. The set of models of $\langle \Omega, \Lambda \rangle$ is denoted by $\mathcal{M}(\langle \Omega, \Lambda \rangle)$.*

In that respect, alignments act as model filters for the ontologies. They select the ontology interpretations which are coherent with the alignments. This allows for transferring information from one ontology to another since reducing the set of models entails more consequences in each aligned ontology.

Example 46 (Model of a network of ontologies)**.** *Hence, a model for the network of ontologies of Figure 8.3 with Δ as defined in Example 43, is $\langle \{m_1, m_2, m_3\}, \Delta \rangle$ built on any models m_1, m_2 and m_3 of ontology o_1, o_2 and o_3 such that $m_3(e_3) \subseteq m_1(b_1) \subseteq m_2(d_2)$, $m_2(c_2) \subseteq m_3(b_3)$ and $m_3(f_3) \subseteq m_1(e_1)$.*

Standard normalisation provides a semantically equivalent network according to this semantics.

Property 24 (Soundness of standard normalisation)**.** *Let $\langle \Omega, \Lambda \rangle$ be a network of ontologies and $\langle \Omega, \overline{\Lambda} \rangle$ its standard normalisation,*

$$\mathcal{M}(\langle \Omega, \Lambda \rangle) = \mathcal{M}(\langle \Omega, \overline{\Lambda} \rangle)$$

Proof. $\mathcal{M}(\langle\Omega,\Lambda\rangle) = \mathcal{M}(\langle\Omega,\overline{\Lambda}\rangle)$ because each model is made of a set of models of the same ontologies Ω and a semantic structure Δ which has to satisfy the same set of correspondences.

More precisely, if $\langle m,\Delta\rangle \in \mathcal{M}(\langle\Omega,\Lambda\rangle)$, then $\forall o,o' \in \Omega, \forall A \in \Lambda(o,o'), \forall \mu \in A, m_o, m_{o'} \models_\Delta \mu$ which means that $\forall o,o' \in \Omega, \forall \mu \in \overline{\lambda}(o,o'), m_o, m_{o'} \models_\Delta \mu$ because either $\Lambda(o,o') = \varnothing$ and then $\overline{\lambda}(o,o')$ does not contain any μ or $\forall \mu \in \overline{\lambda}(o,o'), \exists A \in \Lambda(o,o')$ such that $\mu \in A$ and thus $m_o, m_{o'} \models_\Delta \mu$. Hence, $\langle m,\Delta\rangle \in \mathcal{M}(\langle\Omega,\overline{\Lambda}\rangle)$

In the reverse direction, if $\langle m,\Delta\rangle \in \mathcal{M}(\langle\Omega,\overline{\Lambda}\rangle)$, then $\forall o,o' \in \Omega$, $\forall \mu \in \overline{\lambda}(o,o')$, $m_o, m_{o'} \models_\Delta \mu$, but $\forall o,o' \in \Omega, \forall A \in \Lambda(o,o'), \forall \mu \in A$, $\mu \in \overline{\lambda}(o,o')$ because $\overline{\lambda}(o,o') = \bigcup_{A\in\Lambda(o,o')} A$, thus $m_o, m_{o'} \models_\Delta \mu$. Hence, $\langle m,\Delta\rangle \in \mathcal{M}(\langle\Omega,\Lambda\rangle)$. □

This justifies the position to only consider normalised networks of ontologies.

We consider an order relation $\ll$ between semantic structures denoting the reinforcement of constraints (more complete relations between domains of interpretations or more disjunctions between domains). The stronger the semantic structure, the less models it accepts.

Definition 74 (Constraint reinforcement)**.** *Given two semantic structures Δ and Δ',*

$$\Delta \ll \Delta' \textit{ iff } \forall m,m', \forall \mu, m, m' \models_{\Delta'} \mu \Rightarrow m, m' \models_\Delta \mu$$

The models of a network and that of its standard normalisation can be compared because they are indexed by the same set of ontologies. However, it is not generally possible to directly compare two sets of models of two networks because the set of ontologies that index them is not the same. Therefore, this is again done up to homomorphism.

Definition 75 (Model inclusion)**.** *Given two networks of ontologies $\langle\Omega,\Lambda\rangle$ and $\langle\Omega',\Lambda'\rangle$, the set of models of the latter is said included in that of the former, and denoted by $\mathcal{M}(\langle\Omega',\Lambda'\rangle) \trianglelefteq \mathcal{M}(\langle\Omega,\Lambda\rangle)$, if and only if there exists a map $h : \Omega \longrightarrow \Omega'$ such that $\forall\langle m',\Delta'\rangle \in \mathcal{M}(\langle\Omega',\Lambda'\rangle)$, $\exists\langle m,\Delta\rangle \in \mathcal{M}(\langle\Omega,\Lambda\rangle)$; $\forall o \in \Omega, m_o = m'_{h(o)}$ and $\Delta \ll \Delta'$.*

Property 25 shows that syntactically subsumed networks of ontologies have more models.

Property 25 (Model antitony)**.** *Let $\langle\Omega,\Lambda\rangle$ and $\langle\Omega',\Lambda'\rangle$ be two networks of ontologies,*

$$\langle\Omega,\Lambda\rangle \sqsubseteq \langle\Omega',\Lambda'\rangle \Rightarrow \mathcal{M}(\langle\Omega',\Lambda'\rangle) \trianglelefteq \mathcal{M}(\langle\Omega,\Lambda\rangle)$$

Proof. $\langle\Omega,\Lambda\rangle \sqsubseteq \langle\Omega',\Lambda'\rangle$ means that $\exists\langle h,k\rangle$; $\forall o \in \Omega, o \subseteq h(o)$ and $\forall o,o' \in \Omega$, $\forall A \in \Lambda(o,o')$, $A \subseteq k(A) \wedge k(A) \in \Lambda'(h(o),h(o'))$. Hence, $\exists\langle h,k\rangle$; $\forall m \in \mathcal{M}(h(o))$, $m \in \mathcal{M}(o)$ and $\forall\langle m,m'\rangle \in \mathcal{M}(h(o)) \times \mathcal{M}(h(o'))$, $\forall\Delta', m, m' \models_{\Delta'} k(A) \Rightarrow m, m' \models_{\Delta'} A$ (because $A \subseteq k(A)$). In addition, because A imposes less constraints on the models than $k(A)$, models may be defined with $\Delta \ll \Delta'$, but in such a case, $m, m' \models_\Delta A$. Thus, there exists a map $h : \Omega \longrightarrow \Omega'$ such that $\forall\langle m',\Delta'\rangle \in \mathcal{M}(\langle\Omega',\Lambda'\rangle)$, $\exists\langle m,\Delta\rangle \in \mathcal{M}(\langle\Omega,\Lambda\rangle)$; $\forall o \in \Omega, m_o = m'_{h(o)}$ and $\Delta \ll \Delta'$. In consequence, $\mathcal{M}(\langle\Omega',\Lambda'\rangle) \trianglelefteq \mathcal{M}(\langle\Omega,\Lambda\rangle)$. □

Property 26 (Downward consistency preservation)**.** *Let $\langle\Omega,\Lambda\rangle$ and $\langle\Omega',\Lambda'\rangle$ be two networks of ontologies, If $\langle\Omega,\Lambda\rangle \sqsubseteq \langle\Omega',\Lambda'\rangle$ and $\langle\Omega',\Lambda'\rangle$ is consistent, then $\langle\Omega,\Lambda\rangle$ is consistent.*

Proof. Straightforward from Property 25, since if $\langle\Omega',\Lambda'\rangle$ is consistent, it has a model, and so does $\langle\Omega,\Lambda\rangle$. □

It is expected that all constraints applying to the semantics are preserved by the syntactic morphisms. However, the converse is not guarantee: if a network preserves the constraints of another then there does not necessarily imply that there exist a syntactic morphism.

10.3 Consistency, consequence and closure

A network of ontologies is consistent if it has a model. By extension, an ontology or an alignment is consistent within a network of ontologies if the network of ontologies is consistent. Hence even if an ontology is consistent when taken in isolation, it may be inconsistent when inserted in a network of ontologies. Moreover, if one of the ontologies in the network is inconsistent, then the network as a whole is inconsistent.

Example 47 (Inconsistency)**.** *A model of the network of ontologies presented in Example 36 according to the interpretation of correspondence of Example 43, retains families of models $\{m_1, m_2, m_3\}$ satisfying all alignments, i.e., in particular, satisfying:*

$$m_3(b_3) \supseteq m_2(c_2)$$
$$m_2(c_2) \supseteq m_1(b_1)$$
$$m_1(b_1) \supseteq m_3(e_3)$$

But, all models m_3 of o_3 must satisfy $m_3(b_3) \cap m_3(c_3) = \varnothing$, $m_3(i) \in m_3(e_3)$, and $m_3(i) \in m_3(c_3)$. Moreover, all models m_2 of o_2 must satisfy $m_2(d_2) \subseteq m_2(c_2)$. Hence, $m_3(b_3) \supseteq m_1(b_1)$, $m_3(b_3) \supseteq m_3(e_3)$ and then $m_3(i) \in m_3(b_3)$, which is contradictory with previous assertions. Thus, there cannot exists such a family of models and there is no model for this network of ontologies.

If the interpretation of Example 45 were retained, there may be models in which $\gamma \circ m_3(i) = \varnothing$.

In this example, taking any of the ontologies with only the alignments which involve them, e.g., $\langle \Omega, \{A_{1,3}, A_{2,3}\}\rangle$, is a consistent network of ontologies. The following examples will also consider the network of ontologies $\langle \Omega', \Lambda\rangle$=$\langle \{o_1, o_2, o'_3\}, \{A_{1,2}, A_{1,3}, A_{2,3}\}\rangle$ such that o'_3 is $o_3 \setminus \{i \sqsubseteq e_3\}$.

So far, we have not defined what it means for a formula to be the consequence of a network. There are two notions of consequences called ω-consequence and α-consequence.

α-consequences are correspondences which are consequences of networks of ontologies [EUZENAT 2007].

Definition 76 (α-Consequence of networks of ontologies)**.** *Given a finite set of ontologies Ω and a finite set of alignments Λ between pairs of ontologies in Ω, a correspondence μ between two ontologies o and o' in Ω is an α-consequence of $\langle \Omega, \Lambda\rangle$ (denoted by $\models_{\Omega,\Lambda} \mu$ or $\langle \Omega, \Lambda\rangle \models \mu$) if and only if for all models $\langle m, \Delta\rangle$ of $\langle \Omega, \Lambda\rangle$, $m_o, m_{o'} \models_\Delta \mu$.*

The set of α-consequences between o and o' is denoted by $Cn^\alpha_{\Omega,\Lambda}(o, o')$. For homogeneity of notation, we will use $Cn^\alpha_{\Omega,\Lambda}(A)$ for denoting $Cn^\alpha_{\Omega,\Lambda}(o, o')$ when $A \in \Lambda(o, o')$. The α-closure of a network of ontologies is its set of α-consequences: the correspondences which are satisfied in all models of the network of ontologies.

From the alignment semantics, it is possible to decide if an alignment is a consequence of another or if the alignment makes the set of ontologies and alignments inconsistent.

Example 48 (α-consequences)**.** *The closure of $A_{1,3}$ in the network of ontology $\langle \Omega', \Lambda\rangle$ of Example 47 is:*

$$Cn^\alpha_{\Omega',\Lambda}(o_1, o'_3) = \left\{ \begin{array}{c} e_1 \geq f_3, b_1 \geq e_3 \\ c_1 \geq f_3, a_1 \geq f_3 \\ a_1 \geq e_3, b_1 \leq b_3 \\ b_1 \leq a_3, b_1 \perp c_3 \\ b_1 \perp d_3, b_1 \perp e_3 \end{array} \right\}$$

but if the network is reduced to the two involved ontologies (o_1 and o'_3) only, the closure would be:

$$Cn^\alpha_{\{o_1,o'_3\},\{A_{1,3}\}}(o_1, o'_3) = \left\{ \begin{array}{c} e_1 \geq f_3, b_1 \geq e_3 \\ c_1 \geq f_3, a_1 \geq f_3 \\ a_1 \geq e_3 \end{array} \right\}$$

It is thus clear that connecting more ontologies provides more information.

According to these definitions, $Cn^\alpha(A) = Cn^\alpha_{\langle\{o,o'\},\{A\}\rangle}(A)$ when $A \in \Lambda(o,o')$. α-consequences of an alignment are defined as the α-consequences of the network made of this alignment and the two ontologies it connects. The α-consequences of a particular alignment are usually larger than the alignment ($\forall A \in \Lambda, A \subseteq Cn^\alpha(A) \subseteq Cn^\alpha_{\Omega,\Lambda}(A)$). If the alignment is not satisfiable, then any correspondence is one of its α-consequences.

Similarly, the ω-consequences of an ontology in a network are formulas that are satisfied in all models of the ontology selected by the network.

Definition 77 (ω-Consequence of an ontology in a network of ontologies)**.** *Given a finite set of ontologies Ω and a finite set of alignments Λ between pairs of ontologies in Ω, a formula δ in the ontology language of $o \in \Omega$ is an ω-consequence of o in $\langle\Omega,\Lambda\rangle$ (denoted by $o \models_{\Omega,\Lambda} \delta$) if and only if for all models $\langle m,\Delta\rangle$ of $\langle\Omega,\Lambda\rangle$, $m_o \models \delta$ (the set of ω-consequences of o is denoted by $Cn^\omega_{\Omega,\Lambda}(o)$).*

The ω-closure of an ontology is the set of its ω-consequences. According to these definitions, $Cn^\omega(o) = Cn^\omega_{\langle\{o\},\varnothing\rangle}(o)$. These ω-consequences are larger than the classical consequences of the ontology ($\forall o \in \Omega, o \subseteq Cn^\omega(o) \subseteq Cn^\omega_{\Omega,\Lambda}(o)$) because they rely on a smaller set of models.

Example 49 (ω-consequences)**.** *The simple consequences of the ontology o'_3 are:*

$$Cn^\omega(o'_3) = \left\{ \begin{array}{cc} b_3 \sqsubseteq a_3, c_3 \sqsubseteq a_3, & g_3 \sqsubseteq b_3 \\ d_3 \sqsubseteq c_3, e_3 \sqsubseteq c_3, & f_3 \sqsubseteq b_3 \\ f_3 \perp c_3, b_3 \perp c_3 & f_3 \sqsubseteq a_3, \\ d_3 \sqsubseteq a_3, e_3 \sqsubseteq a_3, & g_3 \perp c_3 \\ d_3 \perp b_3, e_3 \perp b_3 & g_3 \sqsubseteq a_3 \end{array} \right\}$$

while within $\langle\Omega',\Lambda\rangle$ of Example 47, there are even more consequences:

$$Cn^\omega_{\Omega',\Lambda}(o'_3) = \left\{ \begin{array}{cc} b_3 \sqsubseteq a_3, c_3 \sqsubseteq a_3, & g_3 \sqsubseteq b_3 \\ d_3 \sqsubseteq c_3, e_3 \sqsubseteq c_3, & f_3 \sqsubseteq b_3 \\ f_3 \perp c_3, b_3 \perp c_3 & f_3 \sqsubseteq a_3, \\ d_3 \sqsubseteq a_3, e_3 \sqsubseteq a_3, & g_3 \perp c_3, \\ d_3 \perp b_3, e_3 \perp b_3 & g_3 \sqsubseteq a_3 \\ b_3 \sqsupseteq e_3 & \end{array} \right\}$$

From the notion of consequence, we introduce semantic morphism.

Definition 78 (Semantic morphism between networks of ontologies)**.** *Given two networks of ontologies, $\langle\Omega,\Lambda\rangle$ and $\langle\Omega',\Lambda'\rangle$, a* semantic morphism *between $\langle\Omega,\Lambda\rangle$ and $\langle\Omega',\Lambda'\rangle$, is $\exists\langle h,k\rangle$, a pair of morphisms: $h : \Omega \longrightarrow \Omega'$ and $k : \Lambda \longrightarrow \Lambda'$ such that $\forall o \in \Omega$, $\exists h(o) \in \Omega'$ and $h(o) \models_{\Omega',\Lambda'} o$ and $\forall A \in \Lambda(o,o')$, $\exists k(A) \in \Lambda'(h(o),h(o'))$ and $k(A) \models_{\Omega',\Lambda'} A$.*

Semantic morphisms can be used to provide a more semantic definition of subsumption.

We can also define the closure of a network of ontologies by the network of ontologies which replaces each ontology by its ω-closure and each alignment by its α-closure:

$$Cn(\langle\Omega,\Lambda\rangle) = \langle\{Cn^\omega_{\Omega,\Lambda}(o)\}_{o\in\Omega}, \{Cn^\alpha_{\Omega,\Lambda}(o,o')\}_{o,o'\in\Omega}\rangle$$

or alternatively:

$$\langle\{\{\delta \mid \forall\langle m,\Delta\rangle \in \mathcal{M}(\langle\Omega,\Lambda\rangle), m_o \models \delta\}\}_{o\in\Omega}, \{\{\mu \mid \forall\langle m,\Delta\rangle \in \mathcal{M}(\langle\Omega,\Lambda\rangle), m_o, m_{o'} \models_\Delta \mu\}\}_{o,o'\in\Omega}\rangle$$

We also use the notation $Cn^\alpha_{\langle\Omega,\Lambda\rangle}$ for $Cn^\alpha_{\Omega,\Lambda}$ and $Cn^\omega_{\langle\Omega,\Lambda\rangle}$ for $Cn^\omega_{\Omega,\Lambda}$.

Example 50 (Full network closure)**.** *Here is the closure of the network of ontologies $\langle\Omega', \Lambda\rangle$ of Example 47 (the first set is the syntactic form corresponding to the alignment or ontology, the second set is what is added by the local closure and the last set what is added by the ω-closure or α-closure):*

$$Cn^{\omega}_{\Omega',\Lambda}(o_1) = Cn^{\omega}(o_1) = \left\{ \begin{array}{l} b_1 \sqsubseteq a_1, c_1 \sqsubseteq a_1, \\ d_1 \sqsubseteq c_1, e_1 \sqsubseteq c_1 \end{array} \right\} \bigcup \left\{ \begin{array}{l} d_1 \sqsubseteq a_1, e_1 \sqsubseteq a_1 \end{array} \right\}$$

$$Cn^{\omega}_{\Omega',\Lambda}(o_2) = Cn^{\omega}(o_2) = \left\{ \begin{array}{l} b_2 \sqsubseteq a_2, c_2 \sqsubseteq a_2, \\ g_2 \sqsubseteq b_2, f_2 \sqsubseteq b_2, \\ d_2 \sqsubseteq c_2, e_2 \sqsubseteq c_2, \end{array} \right\} \bigcup \left\{ \begin{array}{l} d_2 \sqsubseteq a_2, e_2 \sqsubseteq a_2, \\ f_2 \sqsubseteq a_2, g_2 \sqsubseteq a_2 \end{array} \right\}$$

$$Cn^{\omega}_{\Omega',\Lambda}(o'_3) = \left\{ \begin{array}{l} b_3 \sqsubseteq a_3, c_3 \sqsubseteq a_3, \\ g_3 \sqsubseteq b_3, d_3 \sqsubseteq c_3, \\ e_3 \sqsubseteq c_3, f_3 \sqsubseteq b_3, \\ b_3 \bot c_3 \end{array} \right\} \bigcup \left\{ \begin{array}{l} f_3 \sqsubseteq a_3, g_3 \sqsubseteq a_3, \\ d_3 \sqsubseteq a_3, e_3 \sqsubseteq a_3, \\ d_3 \bot b_3, e_3 \bot b_3, \\ d_3 \bot f_3, d_3 \bot g_3, \\ e_3 \bot f_3, e_3 \bot g_3, \\ f_3 \bot c_3, g_3 \bot c_3 \end{array} \right\} \bigcup \{b_3 \sqsupseteq e_3\}$$

The network does not introduce new assertions in the two first ontologies, but the last one receives a new assertion. Similarly, for alignments, their local closure does not provide new correspondences, but the α-closure becomes larger. These alignment closures are:

$$Cn^{\alpha}_{\Omega',\Lambda}(o_1, o_2) = \left\{ \begin{array}{l} b_1 \leq d_2 \end{array} \right\} \bigcup \left\{ \begin{array}{l} b_1 \leq a_2, \quad b_1 \leq c_2 \end{array} \right\}$$

$$Cn^{\alpha}_{\Omega',\Lambda}(o_2, o'_3) = \left\{ \begin{array}{l} c_2 \leq b_3 \end{array} \right\} \bigcup \left\{ \begin{array}{lll} c_2 \leq a_3, & d_2 \leq a_3, & e_2 \leq a_3 \\ d_2 \leq b_3, & e_2 \leq b_3, & \\ c_2 \bot c_3, & c_2 \bot d_3, & c_2 \bot e_3 \\ d_2 \bot c_3, & d_2 \bot d_3, & d_2 \bot e_3 \\ e_2 \bot c_3, & e_2 \bot d_3, & e_2 \bot e_3 \\ d_2 \geq e_3, & c_2 \geq e_3, & a_2 \geq e_3 \end{array} \right\}$$

$$Cn^{\alpha}_{\Omega',\Lambda}(o_1, o'_3) = \left\{ \begin{array}{l} e_1 \geq f_3, \\ b_1 \geq e_3 \end{array} \right\} \bigcup \left\{ \begin{array}{lll} c_1 \geq f_3, & a_1 \geq f_3, & a_1 \geq e_3, \\ b_1 \bot c_3, & b_1 \bot d_3, & b_1 \bot e_3, \\ b_1 \leq b_3, & b_1 \leq a_3 & \end{array} \right\}$$

Such a representation is highly redundant as closures usually are.

The closure of a network of ontologies may introduce non empty alignments between ontologies which were not previously connected or empty. This is possible because constraints do not come locally from the alignment but from the whole network of ontologies. Such a formalism contributes to the definition of the meaning of alignments: it describes what are the consequences of ontologies with alignments, i.e., what can be deduced by an agent.

Property 27. *Cn is a closure operation on normalised networks of ontologies*[1].

This property can rely on a lemma mirroring Property 25.

Lemma 28 (Consequence isotony)**.** *Let $\langle\Omega, \Lambda\rangle$ and $\langle\Omega', \Lambda'\rangle$ be two networks of ontologies,*

$$\langle\Omega, \Lambda\rangle \sqsubseteq \langle\Omega', \Lambda'\rangle \Rightarrow Cn(\langle\Omega, \Lambda\rangle) \sqsubseteq Cn(\langle\Omega', \Lambda'\rangle)$$

[1] A closure operation Cn in a set S satisfies three properties: $\forall X, Y \in S : X \subseteq Cn(X)$, $Cn(X) = Cn(Cn(X))$, and $X \subseteq Y \Rightarrow Cn(X) \subseteq Cn(Y)$.

Proof of Lemma 28. From Property 25, we know that there exists $h : \Omega \to \Omega'$ and that $\forall \langle m', \Delta' \rangle \in \mathcal{M}(\langle \Omega', \Lambda' \rangle)$, $\exists \langle m, \Delta \rangle \in \mathcal{M}(\langle \Omega, \Lambda \rangle)$; $\forall o \in \Omega, m_o = m'_{h(o)}$ and $\Delta \ll \Delta'$. Hence, $\exists h : \Omega \to \Omega'$ such that $\forall o \in \Omega$, each model of $h(o)$ is a model of o, so $\{m_o \mid \langle m, \Delta \rangle \in \mathcal{M}(\langle \Omega, \Lambda \rangle)\} \supseteq \{m'_{h(o)} \mid \langle m', \Delta' \rangle \in \mathcal{M}(\langle \Omega', \Lambda' \rangle)\}$. This entails $Cn^{\omega}_{\Omega,\Lambda}(o) \subseteq Cn^{\omega}_{\Omega',\Lambda'}(h(o))$. This also means that $\forall o, o' \in \Omega$, each pair of models $\langle m'_{h(o)}, m'_{h(o')} \rangle$ of $\langle h(o), h(o') \rangle$ is also a pair of models of $\langle o, o' \rangle$. Considering $\mu \in Cn^{\alpha}_{\Omega,\Lambda}(o, o')$, either $\forall \langle m', \Delta' \rangle \in \mathcal{M}(\langle \Omega', \Lambda' \rangle)$, $m'_{h(o)}, m'_{h(o')} \models_{\Delta'} \mu$ and then $\mu \in Cn^{\alpha}_{\Omega',\Lambda'}(h(o), h(o'))$, or $\exists \langle m', \Delta' \rangle \in \mathcal{M}(\langle \Omega', \Lambda' \rangle)$; $m'_{h(o)}, m'_{h(o')} \not\models_{\Delta'} \mu$. In this latter case, $\exists \langle m, \Delta' \rangle \in \mathcal{M}(\langle \Omega, \Lambda \rangle)$, such that $m_o = m'_{h(o)}$. This is a model of $\langle \Omega, \Lambda \rangle$ because this network put less constraints on Δ than $\langle \Omega', \Lambda' \rangle$. But this contradicts the hypothesis that $\mu \in Cn^{\alpha}_{\Omega,\Lambda}(o, o')$. Hence, $Cn^{\alpha}_{\Omega,\Lambda}(o, o') \subseteq Cn^{\alpha}_{\Omega',\Lambda'}(h(o), h(o'))$, so $Cn(\langle \Omega, \Lambda \rangle) \sqsubseteq Cn(\langle \Omega', \Lambda' \rangle)$. □

Proof of Property 27. $\forall \langle \Omega, \Lambda \rangle$ and $\langle \Omega', \Lambda' \rangle$ normalised networks of ontologies:

- $\langle \Omega, \Lambda \rangle \sqsubseteq Cn(\langle \Omega, \Lambda \rangle)$, because $\forall \langle h, k \rangle$ such that $\forall o, o' \in \Omega$, $h(o) = Cn^{\omega}_{\Omega,\Lambda}(o)$ and $k(\lambda(o, o')) = Cn^{\alpha}_{\Omega,\Lambda}(o, o')$, (i) $o \subseteq Cn^{\omega}_{\Omega,\Lambda}(o)$, because $\forall \delta \in o, \forall \langle m, \Delta \rangle \in \mathcal{M}(\langle \Omega, \Lambda \rangle), m_o \models o$, hence $m_o \models \delta$, so $\delta \in Cn^{\omega}_{\Omega,\Lambda}(o)$; and (ii) $\lambda(o, o') \subseteq Cn^{\alpha}_{\Omega,\Lambda}(o, o')$, because $\forall \mu \in \lambda(o, o'), \forall \langle m, \Delta \rangle \in \mathcal{M}(\langle \Omega, \Lambda \rangle)$, $m_o, m_{o'} \models_{\Delta} \mu$, hence $\mu \in Cn^{\alpha}_{\Omega,\Lambda}(\lambda(o, o'))$.
- $\forall \langle m, \Delta \rangle \in \mathcal{M}(\langle \Omega, \Lambda \rangle)$, (i) $\forall o \in \Omega, m_o \in \mathcal{M}(o)$ and $\forall o, o' \in \Omega, m_o, m_{o'} \models_{\Delta} \lambda(o, o')$, (ii) $\forall o \in \Omega$, $m_o \in \mathcal{M}(Cn^{\omega}_{\Omega,\Lambda}(o))$, and (iii) $\forall o, o' \in \Omega$, $m_o, m_{o'} \models_{\Delta} Cn^{\alpha}_{\Omega,\Lambda}(o, o')$ (the two latter assertions because the closure contains elements true in all models). In consequence, $\langle m, \Delta \rangle \in \mathcal{M}(Cn(\langle \Omega, \Lambda \rangle))$, which means that $\mathcal{M}(\langle \Omega, \Lambda \rangle) \subseteq \mathcal{M}(Cn(\langle \Omega, \Lambda \rangle))$, and thus $Cn(\langle \Omega, \Lambda \rangle) \sqsupseteq Cn(Cn(\langle \Omega, \Lambda \rangle))$ (less models means more consequences). By the first clause, $Cn(\langle \Omega, \Lambda \rangle) \sqsubseteq Cn(Cn(\langle \Omega, \Lambda \rangle))$, so, $Cn(\langle \Omega, \Lambda \rangle) \equiv Cn(Cn(\langle \Omega, \Lambda \rangle))$.
- Consequence isotony is proved by Lemma 28

□

Example 51. *Figure 10.1 displays an extreme example of a network. This network is inconsistent in the interpretation of Example 43, though none of its ontologies nor alignments is inconsistent. The inconsistency manifests itself by starting with the network without one of the correspondences and revising it by this correspondence. It can be only solved by suppressing one of the* other *correspondences.*

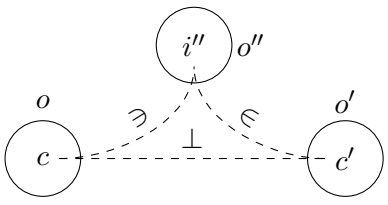


Figure 10.1: Globally inconsistent alignment pattern.

10.4 Local models from the standpoint of an ontology

This definition coincides with a coherent model of the world in which all models satisfy all alignments. This is the standpoint of an omniscient observer and it corresponds to the global knowledge of a distributed system as defined in [FAGIN et al. 1995].

However, if one ontology is inconsistent then the network of ontologies has no model. Therefore, even agents not connected to the inconsistent ontology cannot compute reasonable models. Moreover, an agent

knowing an ontology and the related alignments would like to use the system by gathering information from its neighbours and considering only the models of this information. Thereby, it would be able to compute consequences through some complete deduction mechanisms. This is important when asking agents to answer queries and corresponds to local knowledge in [FAGIN et al. 1995]. This is the knowledge an agent can achieve by communicating only with the agents it is connected to in a network of ontologies.

From that standpoint, there can be several ways to select the acceptable models given the distributed system (here, $X \models_\Delta \lambda(o,o')$ is to be interpreted as $\forall A \in \Lambda(o,o'), X \models_\Delta A$):

$$\mathcal{M}^{0}_{\Omega,\Lambda}(o) = \mathcal{M}(o)$$
$$\mathcal{M}^{\exists}_{\Omega,\Lambda}(o) = \{m \in \mathcal{M}(o); \exists\Delta; \forall o' \in \Omega, \exists m' \in \mathcal{M}(o'); m, m' \models_\Delta \lambda(o,o')\}$$
$$\mathcal{M}^{\exists*}_{\Omega,\Lambda}(o) = \{m \in \mathcal{M}(o); \exists\Delta; \forall o' \in \Omega, \exists m' \in \mathcal{M}^{\exists*}_{\Omega,\Lambda}(o'); m, m' \models_\Delta \lambda(o,o')\}$$
$$\mathcal{M}^{\uparrow\exists}_{\Omega,\Lambda}(o) = \{m \in \mathcal{M}(o); \forall o' \in \Omega, \exists\langle\vec{m},\Delta\rangle \in \mathcal{M}(\langle\Omega,\Lambda\rangle); \vec{m}_o = m\}$$
$$\mathcal{M}^{\forall*}_{\Omega,\Lambda}(o) = \{m \in \mathcal{M}(o); \exists\Delta; \forall o' \in \Omega, \forall m' \in \mathcal{M}^{\forall*}_{\Omega,\Lambda}(o'); m, m' \models_\Delta \lambda(o,o')\}$$
$$\mathcal{M}^{\forall}_{\Omega,\Lambda}(o) = \{m \in \mathcal{M}(o); \exists\Delta; \forall o' \in \Omega, \forall m' \in \mathcal{M}(o'); m, m' \models_\Delta \lambda(o,o')\}$$
$$\mathcal{M}^{\uparrow\forall}_{\Omega,\Lambda}(o) = \{m \in \mathcal{M}(o); \forall o' \in \Omega, \forall\langle\vec{m},\Delta\rangle \in \mathcal{M}(\langle\Omega,\Lambda\rangle); \vec{m}_o = m\}$$

These approaches have been ordered from the more optimistic to the more cautious. $\mathcal{M}^{\exists}_{\Omega,\Lambda}$ selects the models that satisfy each alignment in at least one model of the connected ontology. $\mathcal{M}^{\forall}_{\Omega,\Lambda}$ is very strong since all alignments must be satisfied by all models of the connected ontologies. $\mathcal{M}^{\exists*}_{\Omega,\Lambda}$ and $\mathcal{M}^{\forall*}_{\Omega,\Lambda}$ are fixed point characterisations that, instead of considering the initial models of the connected agents, consider their selected models by the same function. This contributes to propagating the constraints to the entire connected components of the network of ontologies. While for $\mathcal{M}^{\exists*}_{\Omega,\Lambda}$ this strengthens the constraints, for $\mathcal{M}^{\forall*}_{\Omega,\Lambda}$, this relaxes them with respect to $\mathcal{M}^{\forall}_{\Omega,\Lambda}$. Here, an inconsistent model is a problem only to related agents and only for versions $\mathcal{M}^{\exists}_{\Omega,\Lambda}$, and $\mathcal{M}^{\exists*}_{\Omega,\Lambda}$, which require the existence of a model for each related ontology. $\mathcal{M}^{\uparrow\exists}_{\Omega,\Lambda}$ and $\mathcal{M}^{\uparrow\forall}_{\Omega,\Lambda}$ are global versions, which consider models of the whole network of ontologies supporting the filtered models. The satisfaction of $\lambda(o,o')$ is entailed by the choice of m and $\vec{m}_{o'}$ in a model of the network.

Each of these options allows for specialising the semantics of ontologies in a network based on the models of networks of ontologies considered above. It is also analogous to the distributed knowledge of the system following [FAGIN et al. 1995].

One can be even more restrictive by considering only a subset of the possible models of each ontology.

When dealing with ontology matching between a pair of ontologies, the matter of semantics between ontologies is not related to the alignment but to the interpretation of the full network of ontologies, for instance, depending on whether one wants to enforce global consistency or not. In this book we will not take a position on such a matter and will only retain the basic interpretation framework provided above.

10.5 Conclusion

Having a semantics for alignments describes what are the consequences of putting ontologies together. So it contributes giving a meaning to the semantic web.

The semantics of networks of ontologies has been used more systematically to define the category of networks of ontologies [EUZENAT 2014].

In addition, the definition of consequence operations on ontologies or alignments may be used for reasoning with these. For instance, it is possible to formally define alignment composition [ZIMMERMANN and EUZENAT 2006; ZIMMERMANN 2008]. Alignment composition syntactically computes a subset of α-consequences. It may even be used for reasoning about networks of ontologies without knowing the ontologies themselves. More precise reasoning may help testing for alignment consistency which, in turn, may

be used for improving matcher results. We have shown that this semantics may also be used for defining semantic evaluation measures for alignments provided by matchers [EUZENAT 2007].

A network of ontology can be seen as an ontology defining a $\models$ relation. It thus can be included within a network of ontologies as an ontology. This model relation may also be used in the definition of query answering from the standpoint of a particular ontology, hence:

$$\mathcal{A}_{\Omega,\Lambda}(\vec{B}, o, P) = \{\sigma|^{\vec{B}}_{\vec{B}} | o \models_{\Omega,\Lambda} \sigma(P)\}$$

In fact, ontologies and alignments may be though of as complementary: it is possible to use alignments for completing ontologies and using ontologies for completing alignments (while satisfying the semantics).

Finally, such a formalism does not describe what the correct alignments are: matching is not a deductive task but an inductive one. The framework is nevertheless particularly useful for deciding if delivered alignments are consistent, i.e., if networks of ontologies have a model or not. Hence, it is useful for specifying what is expected from matching algorithms and how they should be designed or evaluated.

10.6 Exercises

Exercise 27 (Network of ontologies)**.** *In addition of the ontology o of Exercise 11, we consider an ontology o' which defines the class* `op:Buch` *and contains the following statements:*

$\langle$`d:Baudelaire`, `o:translated`, `d:Confessions`$\rangle$ $\langle$`d:DeQuincey`, `o:wrote`, `d:Confessions`$\rangle$

and o'' which defines the class `opp:Roman` *and contain the following statements:*

$\langle$`d:Confessions`, `rdf:type`, `opp:Roman`$\rangle$ $\langle$`d:Musset`, `o:translated`, `d:Confessions`$\rangle$

They are related together by the following three alignments:

- $A_{o,o'} = \{\langle$`o:Literature`, $\equiv$, `op:Buch`$\rangle\}$
- $A_{o',o''} = \{\langle$`op:Buch`, $\sqsubseteq$, `opp:Roman`$\rangle\}$
- $A_{o'',o} = \{\langle$`opp:Roman`, $\equiv$, `o:Novel`$\rangle\}$

So that we have a network of ontology $\langle\{o, o', o''\}, \{A_{o,o'}, A_{o',o''}, A_{o'',o}\}\rangle$.

1. *Do you think that this network of ontologies is well designed? Why?*
2. *Is this network consistent? Provide a model for this network of ontologies.*
3. *Provide the constraints that the alignments impose on models.*
4. *What does this entail for the class (*`rdf:type`*) of* `d:Confessions` *and* `d:TheRaven` *at o in this network?*

Chapter 11

Equalising semantics for alignments

The usual way of providing a semantics for related conceptual systems is through modal logic of knowledge and belief [FAGIN et al. 1995; WOOLDRIDGE 2000]. In the line of the work on data integration, we only give a first-order model theoretic semantics. It depends on the semantics of ontologies but does not interfere with it. In fact, given a set of ontologies and a set of alignments between them, we can evaluate the semantics of the whole system in terms of the semantics of each individual ontology.

The main problem arising is the non compatibility of the domains of interpretation. Given several ontologies, it is possible to consider different positions with regard to the domain of interpretation:

- For all these ontologies, the domain of interpretation D is unique. This approach is useful when ontologies describe a set of well defined entities, like the set of files shared in a peer-to-peer system. This approach has been taken in [CALVANESE, DE GIACOMO, and LENZERINI 2002; CALVANESE, DE GIACOMO, et al. 2004].
- For each ontology o, the domain D_o may be different. Domains are related with the help of domain relations $r_{o,o'}$ which map elements of D_o to corresponding elements of domain $D_{o'}$. This approach is used in [GHIDINI and SERAFINI 1998; BORGIDA and SERAFINI 2003].
- There is no constraint on the domain of interpretation of ontologies. This is the assumption that will be considered here. For dealing with this assumption, we use a universal domain U, that may be defined as the union of all the domains under consideration, and an equalising function γ or rather a set of equalising functions: $\gamma_o : D_o \longrightarrow U$.

[ZIMMERMANN and EUZENAT 2006] considers the implications of these three models. Here, because the models of various ontologies can have different interpretation domains, we use the notion of an equalising function, which helps make these domains commensurate.

Definition 79 (Equilising function)**.** *Given a family of interpretations $\langle I_o, D_o\rangle_{o\in\Omega}$ of a set of ontologies Ω, an equalising function for $\langle I_o, D_o\rangle_{o\in\Omega}$ is a family of functions $\gamma = (\gamma_o : D_o \longrightarrow U)_{o\in\Omega}$ from the ontology*

Figure 11.1: Equalising semantics.

domains of interpretation to a global domain of interpretation U. The set of all equalising functions is called Γ.

When it is unambiguous, we will use γ as a function. The goal of this γ function is only to be able to (theoretically) compare elements of the domain of interpretation. It is simpler than the use of domain relations in distributed first-order logics [GHIDINI and SERAFINI 1998] in the sense that there is one function per domain instead of relations for each pair of domains.

The equalising functions can be different for each ontology. This means, in particular, that even if two ontologies are interpreted over the same domain of interpretation, it is not compulsory that the equalising function maps their elements to the same element of U, though it remains possible. This allows for a loose coupling of the interpretations.

The relations used in correspondences do not necessarily belong to the ontology languages. As such, they do not have to be interpreted by the ontology semantics. Therefore, we have to provide semantics for them.

Definition 80 (Interpretation of alignment relations)**.** *Given $r \in \Theta$ an alignment relation and U a global domain of interpretation, r is interpreted as a binary relation over U, i.e., $r^U \subseteq U \times U$.*

The definition of correspondence satisfiability relies on γ and the interpretation of relations. It requires that in the equalised models, the correspondences are satisfied.

Definition 81 (Satisfied correspondence)**.** *A correspondence $c = \langle e, e', r\rangle$ is satisfied for an equalising function γ by two models m, m' of o, o' if and only if*

$$\langle \gamma_o(m(e)), \gamma_{o'}(m'(e'))\rangle \in r^U$$

This is denoted as $m, m' \models_\gamma c$.

This definition may be reduced to that of Example 43 if either γ is constrained to be the identity function or $\gamma_o \cdot m \in \mathcal{M}(o)$ and $\gamma_{o'} \cdot m' \in \mathcal{M}(o')$.

Definition 82 (Satisfied alignment)**.** *An alignment A is satisfied for an equalising function γ by two models m, m' of o, o' if and only if all its correspondences are satisfied for γ by m and m'. This is denoted as $m, m' \models_\gamma A$.*

This is useful for defining the classical notions of validity and satisfiability.

Definition 83 (Alignment validity)**.** *An alignment A of two ontologies o and o' is said to be valid if and only if*

$$\forall m \in \mathcal{M}(o), \forall m' \in \mathcal{M}(o'), \forall \gamma \in \Gamma, m, m' \models_\gamma A$$

This is denoted as $\models A$.

From the practical perspective, this is not a very useful definition since unless the ontologies have no models, it will be very difficult to find valid alignments. A relaxed definition could consider the validity that, given an equalising function, describes how domains are related. Valid alignments are direct logical consequences of the two ontologies. Thus they do not provide additional information than what is already in these ontologies. Satisfiable alignments offer more information.

Definition 84 (Satisfiable alignment)**.** *An alignment A of two ontologies o and o' is said to be satisfiable if and only if*

$$\exists m \in \mathcal{M}(o), \exists m' \in \mathcal{M}(o'), \exists \gamma \in \Gamma; m, m' \models_\gamma A$$

Thus, an alignment is satisfiable if there are models of the ontologies that can be combined in such a way that this alignment makes sense. The satisfiable set of alignments is far larger than the set of valid ones. Again, one can define γ-satisfiable alignments, i.e., alignments satisfiable for a given equalising function.

Given an alignment between two ontologies, the semantics of the aligned ontologies can be defined as follows.

Definition 85 (Models of aligned ontologies)**.** *Given two ontologies o and o' and an alignment A between these ontologies, a model m'' of these ontologies aligned by A is a pair $\langle m, m' \rangle \in \mathcal{M}(o) \times \mathcal{M}(o')$ such that there exists $\gamma \in \Gamma$, such that $m, m' \models_\gamma A$.*

This provides the necessary definitions for defining a models of networks of ontologies.

Definition 86 (Models of networks of ontologies)**.** *Given a finite set of n ontologies Ω and a finite set of alignments Λ between pairs of ontologies in Ω, a model of the network of ontologies $\langle \Omega, \Lambda \rangle$ is a n-uple of models $\langle m_1 \dots m_n \rangle \in \mathcal{M}(o_1) \times \dots \mathcal{M}(o_n)$, such that there exists $\gamma \in \Gamma$, such that for each alignment $A \in \Lambda(o_i, o_j)$, A is satisfied by $\langle m_i, m_j, \gamma \rangle$.*

A definition of acceptable models for an ontology, corresponding to a refinment $\mathcal{M}^2_{\Omega,\Lambda}$ in which the equalising function appears, is given as follows:

Definition 87 (Models of an ontology modulo alignments)**.** *Given a network of ontologies $\langle \Omega, \Lambda \rangle$, the models of $o \in \Omega$ modulo Λ are those models of o, such that for each ontology o' there exists a model that satisfies all elements of Λ between o and o':*

$$\mathcal{M}^4_{\Omega,\Lambda}(o) = \{m \in \mathcal{M}(o); \forall o' \in \Omega, \exists \gamma \in \Gamma; \exists m' \in \mathcal{M}^4_{\Omega,\Lambda}(o'); m, m' \models_\gamma \lambda(o, o')\}$$

11.1 Conclusion

The equalising semantics for alignments and networks of ontologies extends the semantic framework presented in Chapter 10. It is quite a natural definition of semantics for reconciling different views of the world in a flexible way.

However, reasoning with such a semantics is a difficult task. We consider in the following chapter the combination of all concepts presented during these lectures for querying the semantic web on principled semantic grounds.

Data interlinking

Data interlinking consists of finding representations of the same entity in two (or more) data sets and linking these representation through the `owl:sameAs` predicate. The resulting triple is called a link and a set of links across the same two data sets a link set. It thus plays a very important role for the web of data (Interlude 1).

This task may be considered as very close to ontology matching (Interlude 4): ontologies are replaced by data sets and alignments by link sets. Like ontology matching this is an inductive task. However, ontology matching is confronted with two ontologies from which it is difficult to extract regularities, though data interlinking faces huge data sets in which regularities can be found. Hence they use different types of techniques. These two activities are rather complementary: link sets may be exploited by extension-based matching techniques and alignments may guide the objects and features to compare within two data sets.

Data interlinking may be divided in two broad types of approaches:

– Similarity-based approaches use a similarity measure to compare entities and, if they are similar enough, they are considered as the same. The similarity either compares different properties of RDF descriptions or project them in a vector space in which similarity is computed.

– Key-based approaches isolate sufficient conditions, called keys or link keys, for two resources to be the same and use them to find same entities. Keys are an extension of relational data base keys which must be associated with an alignment if the data sources do not use the same ontologies [SYMEONIDOU et al. 2014]. Link keys combine in the same structure an alignment and a key across it [ATENCIA, DAVID, et al. 2014].

Both types of approaches may use machine learning or statistical [CAPADISLI et al. 2015] techniques in order to induce "linkage rules" exploiting similarity or link keys. If a sample of links is provided, supervised methods may be used to learn them; otherwise algorithms have to resort on a measure of the quality of the generated links [ATENCIA, DAVID, et al. 2014].

Linkage rules are used to generate the actual links. Link keys may be translated into SPARQL queries. Silk can express and process similarities to generate links against large data sets [VOLZ et al. 2009]. LIMES also uses similarities in metric spaces and focusses on exploiting them efficiently [NGOMO and AUER 2011]. Linkage rules have also been expressed in probabilistic Datalog [AL-BAKRI et al. 2016]. Hence, reasoning can be interleaved with data interlinking.

Chapter 12

Application: Distributed query evaluation

Networks of ontologies allow for connecting ontologies together in a meaningful way. Hence, they can help interpreting information from one ontology into another ontology. In practice, this may be performed in many different ways. Here we illustrate this in a particular type of systems: semantic peer-to-peer systems.

12.1 Semantic peer-to-peer systems

The semantic web is a set of data sources providing information either by answering queries or by offering RDF graphs or snippets. These may not be networks of ontologies because, they may share the same ontologies and not the same data. They are more alike what we call a semantic peer-to-peer network.

A peer-to-peer sharing network is a set of nodes each one aware of other nodes called its acquaintances able to share some resources, e.g., documents. They become semantic when their resources are described through semantic web technologies.

Figure 12.1 illustrates such a system.

It is more formally defined in Definition 88. This definition is very close to other definitions (usually not using the concept of network of ontologies), such as the one developed in [CUDRÉ-MAUROUX 2008].

Definition 88 (Semantic peer-to-peer system)**.** *A semantic peer-to-peer system is a tuple* $\langle \Pi, \Omega, \Lambda, \Psi, \alpha, \omega, \psi, \rho \rangle$*, such that:*

- Π *is a set of peers;*
- $\langle \Omega, \Lambda \rangle$ *is a normalised network of ontologies;*
- Ψ *is a set of resources identified by their IRIs disjoint from those of* Ω*;*
- $\alpha : \Pi \to 2^{\Pi}$ *is a function associating to each peer its acquaintances;*
- $\omega : \Pi \to \Omega$ *is a function associating to each peer its ontology;*
- $\psi : \Pi \to 2^{\Psi}$ *is a function associating to a peer the resources it knows;*
- $\rho : \Pi \to RDF$ *is a function associating to each peer its data as an RDF graph made out of* $(\psi(p) \times \{rdf : type\} \times \omega(p)) \cup (\psi(p) \times \omega(p) \times \psi(p))$

It is possible to complexify this schema by having a peer using several ontologies at once ($\omega : \Pi \to 2^{\Omega}$) and/or by considering that each peer knows a different part of the network of ontologies (it is not public: $\eta : \Pi \to 2^{\Omega} \times 2^{\Lambda}$). But, in first approximation, we only consider Definition 88. There is something more about a peer-to-peer semantic system which is the way to acquire knowledge, e.g., its query language. However, we will not consider here.

Now that we know what is the semantics of networks of ontologies, the semantics of an RDF graph and that of SPARQL query, how can we define the semantics of a semantic peer-to-peer system?

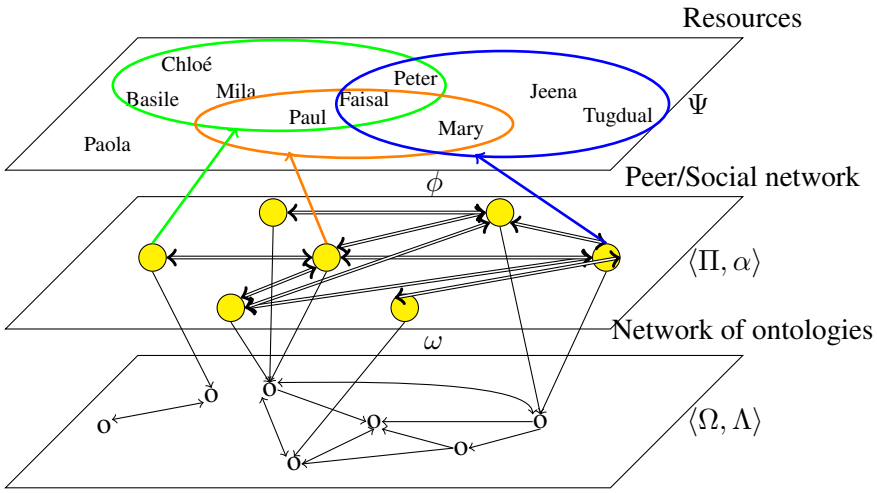


Figure 12.1: A semantic peer-to-peer system.

12.2 The semantics of semantic peer-to-peer systems

With respect to the previous semantics, there is a common dataset, Ψ which facilitates the expression of the semantics because peers are only interested in interpretations for which the domain is (congruent to) Ψ.

12.2.1 What the system knows?

The system has absolute knowledge. It has access to all ontologies and all data graphs in the system. Hence it can put all these graphs together and what it knows is:

$$\bigcup_{p\in\Pi}\{\delta;\omega(p),\rho(p)\models_{\Omega,\Lambda}\delta\}$$

which should correspond to the consequences of the network of ontology amplified by each peer's data graph:

$$Cn(\langle\{o\in\Omega; o\cup\bigcup_{p\in\Pi;\omega(p)=o}\rho(p)\},\Lambda\rangle)$$

It is noteworthy that this is independent from the topology of the peer-to-peer system (the acquaintances).

12.2.2 What a peer knows by itself?

What a peer knows in isolation depends only on its ontology and data graph:

$$\{\delta\in\psi(p);\omega(p),\rho(p)\models\delta\}$$

However, since the network of ontologies is public, it can be used for having more informtation:

$$\{\delta\in\psi(p);\omega(p),\rho(p)\models_{\Omega,\Lambda}\delta\}$$

So, if a peer is seeking for the answer to a query P, it will be:

$$\mathcal{A}^{\pi}(\vec{B},p,P)=\mathcal{A}_{\Omega,\Lambda}(\vec{B},\omega(p)\cup\rho(p),P)$$

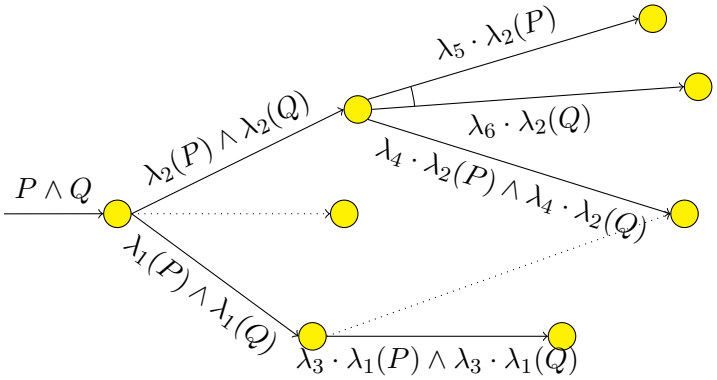


Figure 12.2: Query evaluation strategy.

12.2.3 What a peer will (eventually) know?

The goal of peer-to-peer systems is to take advantage of other peer's knowledge for gathering more information. The peer is limited by its acquaintances in addition to the connectivity of the network of ontologies. In particular, given a particular topology of the peer network, it can only acquire information which are accessible to him.

It can acquire information about resources that it did not new and do it based on ontologies it does not master. The answer to the same query in this context will be:

$$\mathcal{A}^{\pi *}(\vec{B}, p, P) = \mathcal{A}^{\pi}(\vec{B}, p, P) \cup \bigcup_{q \in \alpha(p)} \mathcal{A}^{\pi *}(\vec{B}, q, \lambda_{\omega(p),\omega(q)}(P))$$

It is limited by a non introduced element: the query language used for expressing P. This definition also uses an undefined notation: the application of an alignment to a query. It assumes here that the query may be translated by the alignment. This is not always the case, so, in first approximation, we assume that the answer is empty if the translation is impossible.

However, peers may develop strategies in order to decompose queries and send fragment of these to peers which can deal with these fragments, i.e., such that $\lambda_{\omega(p),\omega(q)}$ is defined on these fragments (see Figure 12.2). Peer may also only send the query to a restricted number of peers if they are confident that the gain in time overweight the (eventual) loss in completeness.

What a peer may know may not necessarily reach what the system knows. That's life.

12.3 Exercises

Exercise 28 (Semantic peer-to-peer system)**.** *Given a peer-to-peer system with peers n_1, n_2 and n_3 all related to each others*[1]*. Peers n_1 and n_2 share the ontology o that has been defined at Question 1 of Exercise 8, n_3 uses the ontology o' defining;*

```
t:Work rdf:type owl:Class.
t:copyrightHolder rdf:type rdf:Property.
t:copyrightHolder rdfs:domain t:Work.
t:year rdf:type rdf:Property.
t:year rdfs:domain t:Work.
t:year rdfs:range xsd:integer.
```

[1]This exercise has not been given at the actual exam.

There exists an alignment A between o and o' containing two correspondences: `t:Work` $\geq$ `m:Livre`, `t:year` $\equiv$ `m:annee`. *Assume that n_1 does not contain instances; n_2 contains:*

```
http://mm.com#a345 m:aecrit http://isbn.org/2070360423.
http://mm.com#a345 m:aecrit http://isbn.org/2070360024.
http://mm.com#a345 m:aecrit http://isbn.org/2070322882.
http://mm.com#a345 foaf:name "Albert".
http://isbn.org/2070360423 rdf:type m:Roman.
http://isbn.org/2070360423 dc:title "La peste".
http://isbn.org/2070360024 rdf:type m:Roman.
http://isbn.org/2070360024 dc:title "L'étranger".
http://isbn.org/2070322882 rdf:type m:Livre.
http://isbn.org/2070322882 dc:title "Le Mythe de Sisyphe".
...
```

and n_3 contains:

```
http://isbn.org/2070360423 t:year 1947.
http://isbn.org/2070322882 t:year 1942.
...
```

1. *Express the alignment A in OWL.*
2. *The peer n_1 would like to evaluate the following query:*

   ```
   SELECT ?t, ?y
   PREFIX ...
   WHERE
     ?x foaf:name "Albert".
     ?x m:aecrit ?l.
     ?l rdf:type m:Roman.
     ?l dc:title ?t.
     OPTIONAL { ?l m:annee ?y. }
   ```

   How is it possible to answer this query by using n_2, n_3 and A?
3. *Provide the answer on the available data.*

12.4 Conclusions

Semantic peer-to-peer systems have been introduced as an application of the concepts which have been considered in this lecture.

It is a very flexible definition. Depending on the languages retained for expressing graphs, ontologies or for transmitting queries between peers one obtains a peer-to-peer system with very different properties ranging from classical peer-to-peer systems to full semantic web technologies (see Table 12.1). We have attempted to show the flexibility of the semantic web technologies, so that simply replacing one language by another does not change the definitions (and hopefully the technology for implementing such systems).

This definition remains static: we consider one semantic peer-to-peer network disregarding its evolution. What characterises such systems is that they occur in time: peers acquire information that they include in their knowledge base (ρ), or meet new acquaintances (α); new peers appears and old ones disconnect (Π); and even ontologies (Ω) and alignments (Λ) evolve. Since, any element of the model may evolve over time, it is useful to be able to track these evolutions and to know how the network, or an individual peer, is supposed to react to such changes.

But, this is another story.

	P2P	PDMS	SW
$\rho(p)$	membership	+relations	RDF
Ω	terminologies	DL-Lite	OWL
Q_L	classes	queries	EDOAL
Θ	equivalence	subsumption	relation algebra
queries	class	conjunctive queries	SPARQL$_{OWL}$

Table 12.1: Different expressivity choices for (semantic) peer-to-peer systems (any combination is possible).

Chapter 13

Conclusion

These lectures led us to define in a very precise way what could be the semantics of the semantic web. The semantic web is made of independent and heterogeneous sources of information available to anyone. We have precised the syntax of *graph* formalisms in which this information is offered and have provided a semantics for this syntax. Then we have considered the definition of the vocabulary (*ontology*) used for expressing this information and again provided the semantics of graphs using this vocabulary language. We have also defined a way to extract information from these sources through *queries* and again, we have defined the syntax and semantics of such queries. Finally, because the sources of information do not depend on the same ontologies we have introduced *alignments* for expressing the relationships between ontologies, and defined their syntax and semantics.

The final example showed that these elements can be put together for designing and describing operating systems.

From there further questions may be asked, such as:

- How to model different styles of systems using the same resources?
- What is the impact of query or ontology languages on what may be known by a peer?
- What is the complexity of answering one type of query in such a system?
- How to design an efficient strategy for extracting as much information as possible?
- How to remain robust to inconsistent ontologies or peers?

All these are research questions...

Part IV

Answers to exercises

RDF

Exercise 1 (RDF assertions)**.** *Consider the four graphs of Figure 2.6.*

1. *Are they all well-formed RDF graphs? Why?*

   Graph (b) is not a well-formed RDF graph because a literal, `"La peste"` *is the subject of a relation (*`rdf:type`*).*

2. *Express the graph of Figure 2.6(a) as a set of triples.*

   ```
   ?x foaf:name "Albert".
   ?x m:aecrit ?l.
   ?l rdf:type m:Roman.
   ?l dc:title "La peste".
   ```

   You will list the sets of literals, IRIs and variables (or blanks) found in this graph. Literals: "Albert", "La peste"
   IRIs: `foaf:name`*,* `m:aecrit`*,* `m:Roman`*,* `rdf:type`*,* `dc:title`
   Variables: `?x`*,* `?l`

3. *Give an informal meaning of this graph or its expression in the predicate calculus*

   Some entity named `"Albert"` *has written a Novel titled* `"La peste"`*.*

   $$\exists x, \exists l, name(x, \text{“}Albert''') \wedge aecrit(x, l) \wedge Roman(l) \wedge title(l, \text{“}Lapeste''')$$

4. *Consider the RDF graphs of Figure 2.6 which are well-formed, do some of them entail others?*

   $$(a) \models (d) \text{ and } (d) \models (a)$$

   Explain why.

   Since (a) *is a strict subgraph of* (d)*, then it is obvious that* $(d) \models (a)$*.*
   For the opposite direction, it is sufficient to see that the node labeled by `?y` *in* (d) *can be projected to the node labeled* `?l` *in* (a) *while projecting the other nodes to those nodes bearing the same label. This projection preserves all the edges of the graph and it is complete, then* $(a) \models (d)$*.*

   In predicate calculus, this is:

   $$\exists x, \exists l, name(x, \text{”}Albert\text{”}) \wedge aecrit(x, l) \wedge Roman(l) \wedge title(l, \text{”}La\ peste\text{”})$$

   is equivalent to:

   $$\exists x, \exists l, name(x, \text{”}Albert\text{”}) \wedge aecrit(x, l) \wedge Roman(l) \wedge title(l, \text{”}La\ peste\text{”}) \wedge \exists y, aecrit(x, y)$$

   which is equivalent to:

   $$\exists x, \exists l, \exists y, name(x, \text{”}Albert\text{”}) \wedge aecrit(x, l) \wedge Roman(l) \wedge title(l, \text{”}La\ peste\text{”}) \wedge aecrit(x, y)$$

Exercise 2 (RDF and assertions)**.** *Consider the two graphs of Figure 2.7 dealing with the expression of social relationships.*

1. *Express the graph of Figure 2.7(b) as a set of triples. You will give the list of litterals, IRIs and variables (or blanks) in this graph.*

$$\langle \texttt{?a}, \texttt{foaf:mbox}, \texttt{"paul@e.at"}\rangle\langle \texttt{?a}, \texttt{worksWith}, \texttt{?b}\rangle$$
$$\langle \texttt{?a}, \texttt{playsTennisWith}, \texttt{?c}\rangle\langle \texttt{?b}, \texttt{marriedWith}, \texttt{?c}\rangle$$
$$\langle \texttt{?b}, \texttt{foaf:mbox}, \texttt{"keith@g.es"}\rangle\langle \texttt{?b}, \texttt{foaf:mbox}, \texttt{"keith@i.com"}\rangle$$

$$\mathcal{L} = \{\texttt{"paul@e.at"}, \texttt{"keith@i.com"}, \texttt{"keith@g.es"}\}$$
$$\mathcal{U} = \{\texttt{worksWith}, \texttt{playsTennisWith}, \texttt{foaf:mbox}, \texttt{marriedWith}\}$$
$$\mathcal{B} = \{\texttt{?a}, \texttt{?b}, \texttt{?c}\}$$

2. *What is, informally, the meaning of the graph of Figure 2.7(b) (tell it in English or French or predicate calculus)?*

   An individual whose email address is "paul@e.at" plays tennis with the spouse of a colleague whose email addresses are "keith@g.es" and "keith@i.com". Or, in predicate calculus:

$$\exists a, \exists b, \exists c; worksWith(a, b) \wedge playsTennisWith(a, c) \wedge marriedWith(b, c) \wedge mbox(a, "paul@e.at") \wedge mbox(b, "keith@g.es") \wedge mbox(b, "keith@i.com")$$

3. *Does one of these graphs entail the other? (explain why)*

   a $\not\models$ b because it is not possible to deduce that $\exists$ x; x marriedWith c and x foaf:mbox keith@i.com

   b $\not\models$ a because it is not possible to deduce worksWith from `foaf:knows`*.*

   Another interesting answer has been given by a student. The following SPARQL query:

```
ASK ?x worksWith ?y. ?y foaf:mbox "keith@i.com".
```

   Succeed for (b) and fails for (a), while

```
ASK ?x foaf:knows ?y.
```

   succeed for (a) and fails for (b). Hence, none of these graphs entail the other.

Exercise 3 (RDF graphs)**.** *Here are the 8 triples of an RDF graph G about writers and their works: (all identifiers correspond in fact to IRIs, _:b is a blank node):*

$$\langle \texttt{d:Poe}, \texttt{o:wrote}, \texttt{d:TheGoldBug}\rangle\ \langle \texttt{d:Baudelaire}, \texttt{o:translated}, \texttt{d:TheGoldBug}\rangle$$
$$\langle \texttt{d:Poe}, \texttt{o:wrote}, \texttt{d:TheRaven}\rangle\ \langle \texttt{d:Mallarmé}, \texttt{o:translated}, \texttt{d:TheRaven}\rangle$$
$$\langle \texttt{d:TheRaven}, \texttt{rdf:type}, \texttt{o:Poem}\rangle\ \langle \texttt{d:Mallarmé}, \texttt{o:wrote}, \texttt{\_:b}\rangle$$
$$\langle \texttt{\_:b}, \texttt{rdf:type}, \texttt{o:Poem}\rangle\ \langle \texttt{d:TheGoldBug}, \texttt{rdf:type}, \texttt{o:Novel}\rangle$$

1. *Draw an RDF graph corresponding to these statements*

2. *Express in English the meaning of these statements.*

   Poe wrote the Poem "The Raven" translated by Mallarmé and the novel "The Gold Bug" translated by Baudelaire. Mallarmé wrote a poem.

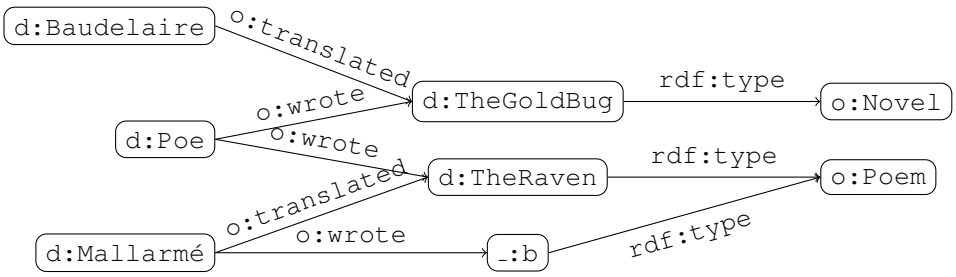


Figure 13.1: The RDF graph G.

Exercise 4 (RDF Manipulation)**.** *Consider the following statements composing the RDF graph G:*

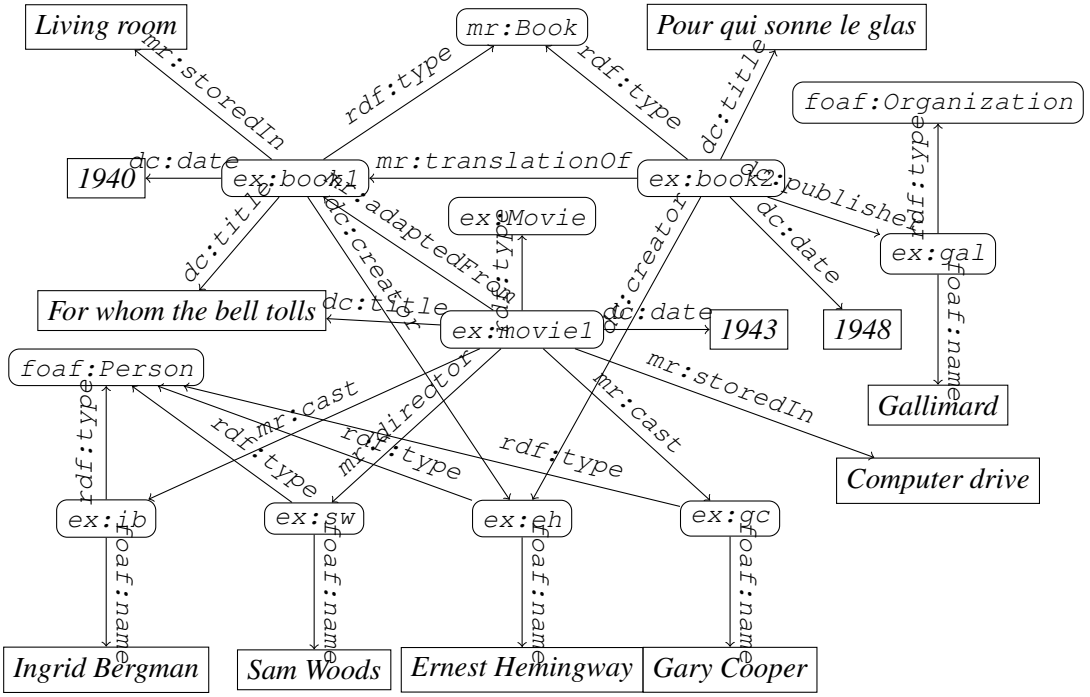
```
ex:book1 rdf:type mr:Book .
ex:book1 dc:title "For whom the bell tolls" .
ex:book1 dc:date 1940 .
ex:book1 dc:creator ex:eh .
ex:book1 mr:storedIn "Living room" .

ex:book2 rdf:type mr:Book .
ex:book2 dc:title "Pour qui sonne le glas" .
ex:book2 mr:translationOf ex:book1 .
ex:book2 dc:creator ex:eh .
ex:book2 dc:date 1948 .
ex:book2 dc:publisher ex:gal .

ex:movie1 rdf:type mr:Movie .
ex:movie1 dc:title "For whom the bell tolls" .
ex:movie1 mr:adaptedFrom ex:book1 .
ex:movie1 mr:director ex:sw .
ex:movie1 mr:cast ex:ib .
ex:movie1 mr:cast ex:gc .
ex:movie1 mr:storedIn "Computer drive" .
ex:movie1 dc:date 1943 .

ex:eh rdf:type foaf:Person .
ex:eh foaf:name "Ernest Hemingway" .

ex:sw rdf:type foaf:Person .
ex:sw foaf:name "Sam Wood .

ex:ib rdf:type foaf:Person .
ex:ib foaf:name "Ingrid Bergman" .

ex:gc rdf:type foaf:Person .
ex:gc foaf:name "Gary Cooper" .

ex:gal rdf:type foaf:Organization .
ex:gal foaf:name "Gallimard" .
```

1. *Which different classes are there in graph G? Why are they classes?*

`mr:Book`, `mr:Movie`, `foaf:Person` and `foaf:Organization`.

They are classes because they are in the range of `rdf:type`. `rdfs:Class` *is the range of* `rdf:type` *from the RDFS Axiomatic triples (Definition 12). Additionnaly, they are also identified by uppercase fragments.*

2. *Express G as a graph in graphical form.*



Consider the following as the RDF graph G' (`_:` are blank identifiers):

```
_:x rdf:type mr:Movie .
_:x mr:adaptedFrom _:y .
_:y dc:creator _:p .
_:p foaf:name "Ernest Hemingway" .
```

3. *Paraphrase the graph G'.*

   There is a movie adapted from some creation of Ernest Hemingway.

4. *Is G' entailed by G? Detail why?*

   Yes, $G \models G'$.

   This is because, for any model $I = \langle\rangle$ of G is such that there exist and extension ι' of ι to the blanks: `_:x`, `_:y`, `_:p` *which assigns them* `ex:movie1`, `ex:book1` *and* `ex:eh`. *This extension satisfies:* $\langle \iota'(\_ : x), \iota'(mr : Movie)\rangle \in I_{EXT}(\iota'(\texttt{rdf:type}))$, $\langle \iota'(\_ : x), \iota'(\_ : y)\rangle \in I_{EXT}(\iota'(\texttt{mr:adaptedFrom}))$, $\langle \iota'(\_ : y), \iota'(\_ : p)\rangle \in I_{EXT}(\iota'(\texttt{dc:creator}))$, *and* $\langle \iota'(\_ : p), \iota'(\text{"}ErnestHemingway\text{"})\rangle \in I_{EXT}(\iota'(\texttt{foaf:name}))$, *because I satisfies:* $\langle \iota(\texttt{ex:movie1}), \iota(mr : Movie)\rangle \in I_{EXT}(\iota(\texttt{rdf:type}))$, $\langle \iota(\texttt{ex:movie1}), \iota(\texttt{ex:book1})\rangle \in I_{EXT}(\iota(\texttt{mr:adaptedFrom}))$, $\langle \iota(\texttt{ex:book1}), \iota(:p)\rangle \in I_{EXT}(\iota(\texttt{dc:creator}))$, *and* $\langle \iota(\texttt{ex:eh}), \iota(\text{"}ErnestHemingway\text{"})\rangle \in I_{EXT}(\iota(\texttt{foaf:name}))$,

Exercise 5 (RDF Entailment)**.** *Consider the graph G of Figure 2.8 (p. 29).*

1. *Explain what it means (in English or French, taking into account the meaning of RDFS vocabulary).*

   CHUs are hospitals. A gyneco-obsetrics service is a maternity. The Belledonne hospital has a pediatry and a gyneco-obsetrics service. The La tronche CHU has a maternity and a radiology service.

2. *Rewrite it as a set of triples.*

```
a:CHU rdfs:subClassOf a:Hospital .
a:Gyneco-Obsetrics rdfs:subClassOf a:Maternity .
b:Belledonne rdf:type a:Hospital .
b:H.Michallon rdf:type a:CHU .
b:Belledone rdfs:label "Belledonne" .
b:H.Michallon rdfs:label "La Tronche" .
b:Belledone a:hasService b:GO .
b:GO rdf:type a:Gyneco-Obsetrics .
b:Belldonne a:hasService b:Pediatry .
b:Pediatry rdf:type a:Pediatry .
b:H.Michallon a:hasService b:Radio .
b:Radio rdf:type a:Radiology .
b:H.Michallon a:hasService b:Maternity .
b:Maternity rdf:type a:Maternity .
```

Given the GRDF graph

P={ *?x* `rdf:type` `a:CHU`. *?x* `a:hasService` *?y*. *?y* `rdf:type` `a:Maternity`. }

3. *Does $G \models_{RDF} P$? (explain how and/or why)*

   Yes. Because the instantiation of P with ?x replaced by `b:H.Michallon` *and ?y replaced by* `b:Maternity` *is a subgraph of G.*

Exercise 6 (RDF Entailment)**.** *We have two graphs coming from two different sources. Consider the graph G_1 coming from the municipality made of the following triples:*

```
a:Pierre o1:father a:Carole .
a:Pierre o1:father a:Kevin .
a:Jacques o1:father a:Jean .
_:b1 o1:father a:Sylvie .
_:b1 o1:father a:William .
a:Jacques o1:father a:Julie .
a:Sven o1:father a:Laurent .
```

```
a:Marie o1:mother a:Carole .
a:Marie o1:mother a:Kevin .
a:Marie o1:mother a:Jean .
a:Stephanie o1:mother a:Sylvie .
a:Stephanie o1:mother a:William .
a:Nabila o1:mother a:Julie.
a:Lucie o1:mother a:Laurent .
```

and G_2 coming from the school made of the following information:

```
a:Carole o2:attendsClass b:4e3.
a:Kevin o2:attendsClass b:6e1 .
a:Sylvie o2:attendsClass b:5e2 .
a:William o2:attendsClass b:5e2 .
a:Julie o2:attendsClass b:5e2 .
a:Laurent o2:attendsClass b:4e3  .
a:Jasmine o2:attendsClass b:5e1 .
```

```
a:Carole rdf:type o2:Female .
a:Kevin rdf:type o2:Male .
a:Sylvie rdf:type o2:Female .
a:William rdf:type o2:Male .
a:Julie rdf:type o2:Female .
a:Laurent rdf:type o2:Male .
a:Jasmine rdf:type o2:Female .
```

1. *Draw these two graphs (together);*

   They are like the graph of Figure 13.2.

2. *In order, to work with these two graphs, we want to answer queries that span through both of them. Consider the following graph Q_1:*

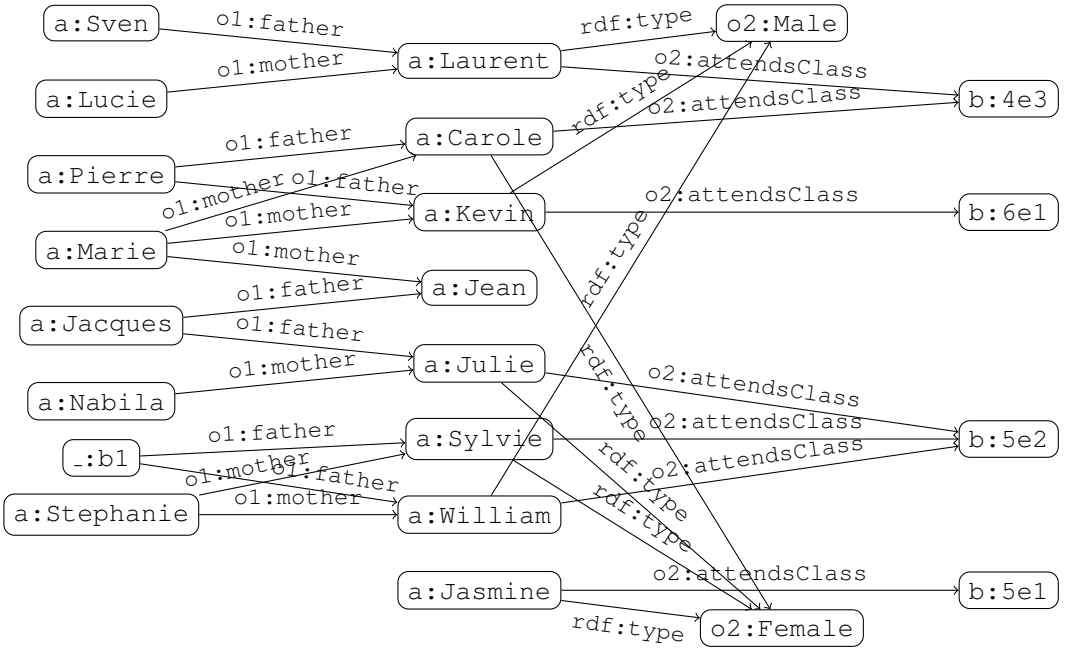


Figure 13.2: RDF graph G_1 and G_2.

```
_:x o2:attendsClass _:w .
_:y o2:attendsClass _:w .
_:x rdf:type o2:Male .
_:y rdf:type o2:Female .
_:z o:parent _:x .
_:z o:parent _:y .
```

Express in English the meaning of Q_1.

Q_1 could be rephrased as "there exist a male and a female sharing at least one parent attending the same class".

Is Q_1 entailed by any of G_1 or G_2? (explain why)

Q_1 is not entailed by any of these graphs, this would require these graphs to contain at least one triple with `o:parent` as predicate and none contains this.

3. *Express the graph Q_2 corresponding to the English: "there exist two people sharing at least one parent attending the same class"?*

   Q_2 can be expressed by the following graph:

```
_:x o2:attendsClass _:w .
_:y o2:attendsClass _:w .
_:z o:parent _:x .
_:z o:parent _:y .
```

   Does $Q_2 \models_{RDF} Q_1$ or $Q_1 \models_{RDF} Q_2$?

   $Q_2 \not\models_{RDF} Q_1$ because there is no subgraph of Q_2 that is an instance of Q_1. In particular, any instance of Q_1 should contain references to `o2:Male` and `o2:Female` which are not present in Q_2 (so neither in any subgraph of Q_2).

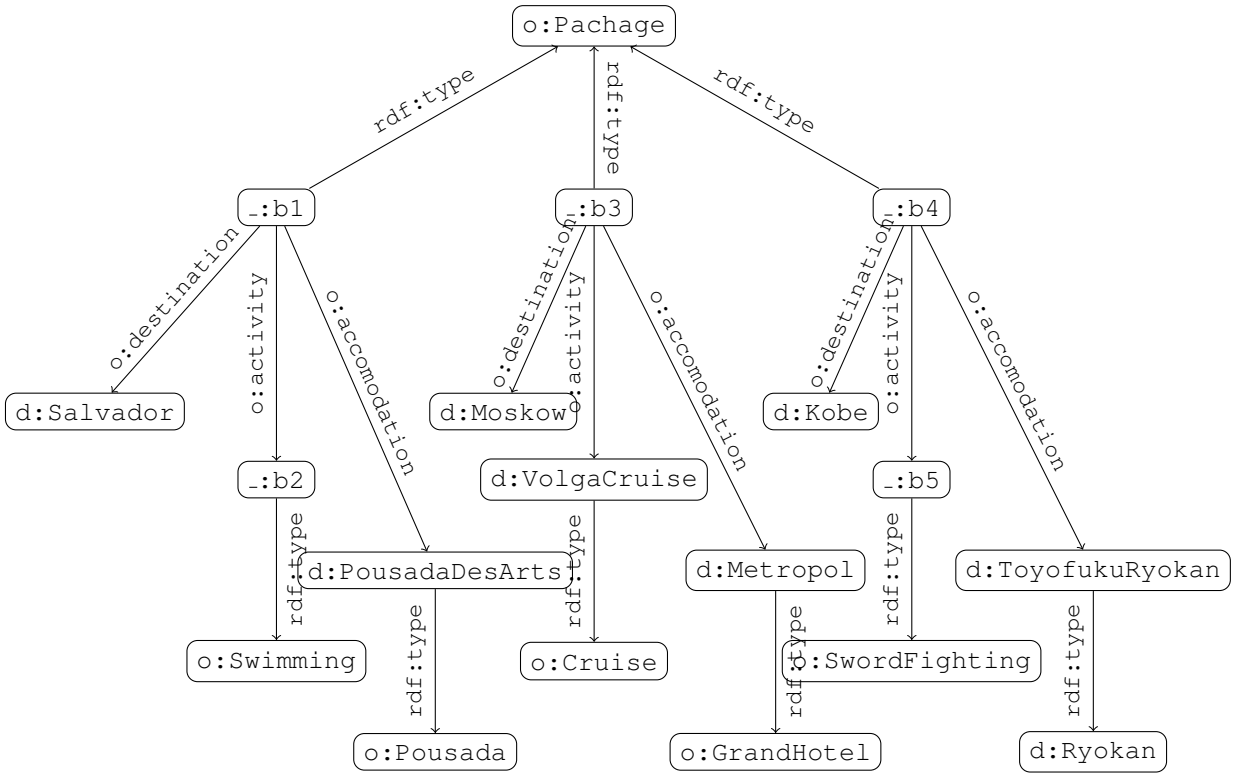


Figure 13.3: RDF graph G.

$Q_1 \models_{RDF} Q_2$ because Q_2 is (and instance of itself and) a subgraph of Q_1, namely the subgraph in which the `rdf:type` information is missing.

Exercise 7 (RDF Interpretation and entailment)**.** *Consider the graph G describing holiday packages:*

```
_:b1 rdf:type o:Package .                d:PousadaDesArts rdf:type o:Pousada .
_:b1 o:destination d:Salvador .          _:b1 o:activity _:b2 .
_:b1 o:accomodation d:PousadaDesArts .   _:b2 rdf:type o:Swimming .

_:b3 rdf:type o:Package .                d:Metropol rdf:type o:GrandHotel .
_:b3 o:destination d:Moskow .            _:b3 o:activity d:VolgaCruise .
_:b3 o:accomodation d:Metropol .         d:VolgaCruise rdf:type o:Cruise .

_:b4 rdf:type o:Package .                d:ToyofukuRyokan rdf:type o:Ryokan .
_:b4 o:destination d:Kobe .              _:b4 o:activity _:b5 .
_:b4 o:accomodation d:ToyofukuRyokan .   _:b5 rdf:type o:SwordFighting .
```

1. *Draw the graph G.*

   The graph of Figure 13.3 corresponds to G.

2. *Define an RDF-interpretation $\mathcal{I}$ of G's vocabulary ($V(G)$).*

$\mathcal{I} = \langle I_R, I_P, I_{EXT}, \iota \rangle$ *such that:*

$$
\begin{aligned}
I_R &\supseteq I_P \cup \{B, C, D\} \\
&\quad \cup \{\iota(\texttt{o:Package}), \iota(\texttt{o:Pousada}), \iota(\texttt{o:GdHotel}), \iota(\texttt{o:Ryokan}), \\
&\qquad \iota(\texttt{o:Swimming}), \iota(\texttt{o:Cruise}), \iota(\texttt{oSwordFighting}), \\
&\qquad \iota(\texttt{d:Salvador}), \iota(\texttt{d:Moskow}), \iota(\texttt{d:Kobe})\} \\
I_P &\supseteq \{\iota(\texttt{rdf:type}), \iota(\texttt{o:destination}), \iota(\texttt{o:accomodation}), \\
&\qquad \iota(\texttt{o:activity})\} \\
I_{EXT}(\iota(\texttt{rdf:type})) &\supseteq \{\langle B, \iota(\texttt{o:Package})\rangle, \langle C, \iota(\texttt{o:Swimming})\rangle, \\
&\qquad \langle \iota(\texttt{d:VolgaCruise}), \iota(\texttt{o:Cruise})\rangle, \\
&\qquad \langle D, \iota(\texttt{o:SwordFighting})\rangle, \\
&\qquad \langle \iota(\texttt{d:PousadaDesArts}), \iota(\texttt{o:Pousada})\rangle, \\
&\qquad \langle \iota(\texttt{d:Metropol}), \iota(\texttt{o:GrandHotel})\rangle, \\
&\qquad \langle \iota(\texttt{d:ToyofukuRyokan}), \iota(\texttt{o:Ryokan})\rangle\} \\
I_{EXT}(\iota(\texttt{o:destination})) &\supseteq \{\langle B, \iota(\texttt{d:Salvador})\rangle, \langle B, \iota(\texttt{d:Moskow})\rangle, \langle B, \iota(\texttt{d:Kobe})\rangle\} \\
I_{EXT}(\iota(\texttt{o:accomodation})) &\supseteq \{\langle B, \iota(\texttt{d:PousadaDesArts})\rangle, \langle B, \iota(\texttt{d:Metropol})\rangle, \\
&\qquad \langle B, \iota(\texttt{d:ToyofukuRyokan})\rangle\} \\
I_{EXT}(\iota(\texttt{o:activity})) &\supseteq \{\langle B, C\rangle, \langle B, \iota(\texttt{d:VolgaCruise})\rangle, \langle B, D\rangle\}
\end{aligned}
$$

It is possible to replace $\iota(\dots)$ by $a, b, \dots$ if it makes you more comfortable. This interpretation is a bit peculiar as it interprets all packages as the same with three destinations, but nothing prohibits this.

3. *Given the following graph H:*

```
_:x rdf:type o:Package .
_:x o:accomodation _:acc .
_:x o:activity _:act .
```

Does your interpretation satisfies H (said otherwise, is $\mathcal{I}$ a model of H)?

Yes, $\mathcal{I}$ a model of H as it is possible to find an extension ι' of ι to $\{\texttt{\_:x}, \texttt{\_:act}, \texttt{\_:acc}\}$ satisfying all triples of H. This is the case, for instance if one takes: $\iota' = \iota \cup \{\langle \texttt{\_:x}, B\rangle, \langle \texttt{\_:act}, C\rangle, \langle \texttt{\_:acc}, \texttt{d:PousadaDesArts}\rangle\}$.

4. *Does $G \models H$? Show it.*

Any model of G is indeed a model of H. For any model $m = \langle I_R, I_P, I_{EXT}, \iota \rangle$ of G, ι can be extended into ι' such that:

$$
\begin{aligned}
\langle \iota'(\texttt{\_:b1}), \iota(\texttt{o:Package})\rangle &\in I_{EXT}(\iota(\texttt{rdf:type})) \\
\langle \iota'(\texttt{\_:b1}), \iota(\texttt{d:PousadaDesArts})\rangle &\in I_{EXT}(\iota(\texttt{o:accomodation})) \\
\langle \iota'(\texttt{\_:b1}), \iota'(\texttt{\_:b2})\rangle &\in I_{EXT}(\iota(\texttt{o:activity}))
\end{aligned}
$$

so it is possible to define the extension ι'' of ι' to $\{\texttt{\_:x}, \texttt{\_:act}, \texttt{\_:acc}\}$ such that: $\iota''(\texttt{\_:x}) = \iota'(\texttt{\_:b1})$, $\iota''(\texttt{\_:act}) = \iota'(\texttt{\_:b2})$, and $\iota''(\texttt{\_:acc}) = \iota(\texttt{d:PousadaDesArts})$. ι'' is an extendion of ι and it satisfies all triples of H, hence, m is a model of H. This can also be achieved by showing that there is an RDF-homomorphism from H to G or that an instance of H is a subgraph of G.

5. *Given the following graph K:*

```
_:y rdf:type o:Package .
_:y o:accomodation _:acc .
_:acc rdf:type o:Local .
_:y o:activity _:act .
_:act rdf:type o:Sport .
```

Does $G \models K$? Tell why.

No, because there is no reference to `o:Sport` in the graph G, hence it is impossible to find an RDF-homomorphism from an instance of K to a subgraph of G as it would need to map the node labelled by `o:Sport` to a node with the same label (see also answer to Question 2).

OWL

Exercise 8 (OWL ontologies)**.**

1. *Describe in OWL (RDF or XML/RDF) the ontology containing the following assertions:*
   - *All authors are persons;*
   - *A book (`m:Livre`) has exactly one year of publication (`m:annee`);*
   - *A novel (`m:Roman`) is a book (`m:Livre`) and a book is a work (`m:Oeuvre`);*
   - *The title (`dc:title`) of a work is a character string (`xsd:string`);*
   - *The relation "a écrit" (`m:aecrit`) relates an author to a work.*

```
<owl:Class rdf:about="#Auteur">
  <rdfs:subClassOf rdf:resource="#Personne"/>
</owl:Class>

<owl:Class rdf:about="#Oeuvre">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="dc:title"/>
      <owl:datatype>&xsd;string</owl:datatype>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:about="#Livre">
  <rdfs:subClassOf rdf:resource="#Oeuvre"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#annee"/>
      <owl:cardinality>1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:about="#Roman">
  <rdfs:subClassOf rdf:resource="#Livre"/>
</owl:Class>

<owl:ObjectProperty rdf:about="#aecrit">
```

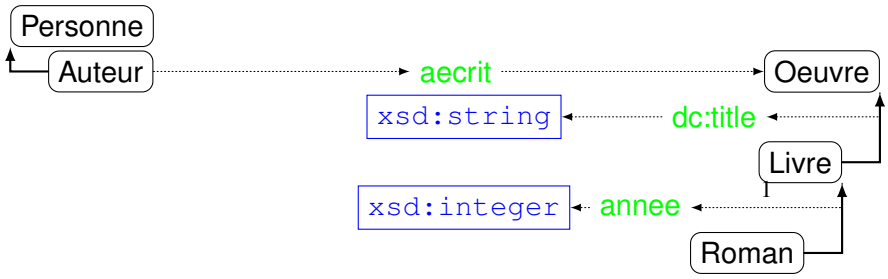


Figure 13.4: OWL ontologies.

```
  <rdfs:domain rdf:resource="#Auteur"/>
  <rdfs:range rdf:resource="#Oeuvre"/>
</owl:ObjectProperty>
```

which corresponds to the diagram of Figure 13.4.

2. *If one associates the graph (a) of Figure 2.6 of Exercise 1 (p26) and the ontology resulting from the previous question, is it possible to deduce the type (`rdf:type`) of `?x`?*

   `?x rdf:type m:Auteur`.

   Can you semantically justify how?

   $I(\texttt{aecrit}) \subseteq I(\texttt{Auteur}) \times I(\texttt{Oeuvre})$
   $\langle I(\texttt{?x}), I(\texttt{?l})\rangle \in I(\texttt{aecrit})$
   hence
   $I(\texttt{?x}) \in I(\texttt{Auteur})$
   thus: `?x rdf:type Auteur` *What else can be deduced?*

   `?l` *has a year of publication:* `?l m:annee ?y`.

3. *Does the use of this ontology for defining the graphs of Figure 2.6 (p26) would change something to the answer to Question 4 of Exercise 1? What?*

   Yes: $(a) \models (c)$ *because* `?l rdf:type m:Roman` $\models$ `?l rdf:type m:Livre`

   and $(d) \models (c)$ *because* $(d) \models (a)$.

Exercise 9 (DL-Lite ontologies)**.** *Consider the ontology O made of the following DL-Lite assertions:*

$$\begin{aligned} \texttt{worksWith} &\sqsubseteq \texttt{foaf:knows} \\ \texttt{playsTennisWith} &\sqsubseteq \texttt{playsSportWith} \\ \texttt{playsSportWith} &\sqsubseteq \texttt{foaf:knows} \\ \texttt{marriedWith} &\sqsubseteq \texttt{foaf:knows} \\ \texttt{Person} &\sqsubseteq \exists \texttt{foaf:mbox} \end{aligned}$$

1. *In which dialect (sublanguage) of DL-Lite is this ontology expressed (DL-Lite$_{core}$, DL-Lite$_{\mathcal{F}}$, DL-Lite$_{\mathcal{R}}$)?*

   This is DL-Lite$_{\mathcal{R}}$ because there are assertions on relations.

2. *Rewrite O in OWL or RDFS.*

```
<owl:ObjectProperty rdf:about="worksWith">
  <rdfs:subPropertyOf rdf:resource="&foaf;knows" />
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="playsTennisWith">
  <rdfs:subPropertyOf rdf:resource="#playsSportWith" />
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="playsSportWith">
  <rdfs:subPropertyOf rdf:resource="&foaf;knows" />
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="marriedWith">
  <rdfs:subPropertyOf rdf:resource="&foaf;knows" />
</owl:ObjectProperty>

<owl:Class rdf:about="Person">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&foaf;mbox" />
      <owl:someValuesFrom rdf:resource="&owl;Thing" />
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

or, for the last one:

```
<owl:Class rdf:about="Person">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&foaf;mbox" />
      <owl:minCardinality>1</owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

3. *In which dialect (sublanguage) of RDFS or OWL is your ontology expressed?*

   This is OWL-Lite. Although, DL-Lite$_\mathcal{R}$ is more expressive than RDFS and non comparable with OWL-Lite, this particular ontology is expressed in OWL-Lite (in which `minCardinality` *constraints are expressible with values of 0 or 1).*

4. *Given the graph of Figure 2.7(b) of Exercise 2 (p27) to which the axioms of O are added. Compute its closure. What is the difference with the partial closure?*

   The closure is:

$$\langle ?a, foaf{:}mbox, "paul@e.at"\rangle\langle ?a, worksWith, ?b\rangle$$
$$\langle ?a, playsTennisWith, ?c\rangle\langle ?b, marriedWith, ?c\rangle$$
$$\langle ?b, foaf{:}mbox, "keith@g.es"\rangle\langle ?b, foaf{:}mbox, "keith@i.com"\rangle$$
$$\langle ?a, foaf{:}knows, ?c\rangle\langle ?a, foaf{:}knows, ?b\rangle$$
$$\langle ?a, playsSportWith, ?c\rangle\langle ?b, foaf{:}knows, ?c\rangle$$

plus the axiomatic triples (like $\langle$`foaf:mbox`, `rdf:type`, `rdf:Property`$\rangle$, $\langle$`rdf:type`, `rdf:type`, `rdf:Property`$\rangle$, $\langle$`rdf:_1`, `rdf:type`, `rdf:Property`$\rangle$*...). The partial closure is the same minus the triples involving* `rdf:_`i *because there is no such IRI in our graph.*

5. *Given the graphs of Figure 2.7 (p27) to which the axioms of O are added, does this change something to the answers given to Question 3 of Exercise 2 if we consider RDFS-entailment? (explain why)*

   Now $b, O \models a$ *because worksWith* $\sqsubseteq$ *foaf:knows.*

Exercise 10 (OWL ontologies)**.**

1. *Provide an ontology satisfied by both graphs of Figure 2.7 of Exercise 2 (p27).*

```
<owl:ObjectProperty rdf:about="&foaf;knows">
  <rdfs:domain rdf:resource="&foaf;Person" />
  <rdfs:range rdf:resource="&foaf;Person" />
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="&foaf;mbox">
  <rdfs:domain rdf:resource="&foaf;Person" />
  <rdfs:range rdf:resource="&xsd;string" />
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="worksWith">
  <rdf:type rdf:resource="&owl;SymmetricProperty" />
  <rdfs:subPropertyOf rdf:resource="&foaf;knows" />
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="playsTennisWith">
  <rdf:type rdf:resource="&owl;SymmetricProperty" />
  <rdfs:subPropertyOf rdf:resource="#playsSportWith" />
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="playsSportWith">
  <rdf:type rdf:resource="&owl;SymmetricProperty" />
  <rdfs:subPropertyOf rdf:resource="&foaf;knows" />
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="marriedWith">
  <rdf:type rdf:resource="&owl;SymmetricProperty" />
  <rdf:type rdf:resource="&owl;FunctionalProperty" />
  <rdf:type rdf:resource="&owl;InverseFunctionalProperty" />
  <rdfs:subPropertyOf rdf:resource="&foaf;knows" />
</owl:ObjectProperty>
```

We would like to express that an email address cannot correspond to more than one person by the property `foaf:mbox`*.*

2. *How to express this in OWL: with a cardinality constraint, a functional property, an inverse functional property or a symmetric property?*

   This is an inverse functional property which states that to an email address corresponds only one image by the inverse of `foaf:mbox`*. This is simply expressed by*[1]*:*

[1] In principle, in OWL 1, this only works on ObjectProperties and `foaf:mbox` is a DataProperty.

```
<owl:InverseFunctionalProperty rdf:about="&foaf;mbox" />
```

It is possible to use:

```
<owl:Class rdf:about="&owl;Thing">

  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty>
          <owl:inverseOf rdf:resource="&foaf;mbox" />
        </owl:ObjectProperty>
      </owl:onProperty>
      <owl:maxCardinality>1</owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

3. *Is it possible to express this constraint in DL-Lite? If yes, how?*

   This is possible in the DL-Lite$_\mathcal{F}$ language:

   ```
   (functional foaf:mbox$^-1$)
   ```

4. *Consider the graphs of Figure 2.7 (p27) to which this constraint is added, does this change something to the answers given to Question 3 of Exercise 2? (explain why)*

   (a) becomes a super-graph of (b), because `?b` *and* `?d` *becomes the same node because they have the* `foaf:mbox "keith@g.es"` *in common and this an inverse functional property. The only difference between both graphs is that* $\langle$`?a`, `foaf:knows`, `?b`$\rangle$ *belongs to (a) and not to (b). As a consequence,* $a, O' \models b$ *(with* O' *the ontology with the inverse functional property constraint), but not the other way around for the same reason as before.*

Exercise 11 (RDFS ontologies)**.** *Consider the RDFS ontology* o *containing, in addition to those of the graph* G *of Exercise 3, the following statements:*

$$\langle \texttt{o:Novel}, \texttt{rdfs:subClassOf}, \texttt{o:Literature}\rangle$$
$$\langle \texttt{o:Poem}, \texttt{rdfs:subClassOf}, \texttt{o:Literature}\rangle$$
$$\langle \texttt{o:translated}, \texttt{rdfs:range}, \texttt{o:Literature}\rangle$$
$$\langle \texttt{o:wrote}, \texttt{rdfs:domain}, \texttt{o:Writer}\rangle$$

1. *Does this allow to conclude that* `d:Poe`, `d:Baudelaire` *or* `d:Mallarmé` *is a* `o:Writer`*? Explain why.*

   The only assertion that would allow to conclude that someone is a `o:Writer` *is the last one related to the domain of the* `o:wrote` *predicate. Nothing allows for inferring triples with the* `o:wrote` *predicate, so the only assertions with it are those asserted. Hence, the only writers are* `d:Poe` *and* `d:Mallarmé`*.*

2. *Can you express in OWL the statement that "anyone who write Literature is a Writer"?*

   The sentence expresses that those who write Literature are Writers, hence Writer is a superclass of the restriction. This can be expressed by creating a class equivalent to the restriction, and subclass of Writer:

```
<owl:Class>
  <owl:equivalentClass>
    <owl:Restriction>
      <owl:onProperty rdf:resource="o:wrote"/>
      <owl:someValueFrom rdf:resource="o:Literature"/>
    </owl:Restriction>
  </owl:equivalentClass>
  <rdfs:subClassOf rdf:resource="o:Writer"/>
</owl:Class>
```

or more briefly:

```
<owl:Restriction>
  <owl:onProperty rdf:resource="o:wrote"/>
  <owl:someValueFrom rdf:resource="o:Literature"/>
  <rdfs:subClassOf rdf:resource="o:Writer"/>
</owl:Restriction>
```

It would have been possible to add a range constraint on the `o:wrote` *predicate so that whatever is written is* $\texttt{o:Literature}(\langle \texttt{o:wrote}, \texttt{rdfs:range}, \texttt{o:Literature}\rangle)$*. However, this is stronger than what was asked since it would have restricted a particular author to write something else than literature.*

Exercise 12 (OWL 2 qualified cardinality restrictions)**.** *OWL 2 introduced qualified cardinality restrictions (*`owl:qualifiedCardinality`*,* `owl:maxQualifiedCardinality`*, and* `owl:minQualifiedCardinality`*, whose interpretation is obtained by extending the* E_C *function of Definition 19:*

$$\begin{aligned} & E_C(\texttt{restriction}(p, \texttt{minQualifiedCardinality}(n, C))) \\ = \ & \{x \in O; |\{\langle x, y\rangle \in E_R(p); y \in E_C(C)\}| \geq n\} \\ & E_C(\texttt{restriction}(p, \texttt{maxQualifiedCardinality}(n, C))) \\ = \ & \{x \in O; |\{\langle x, y\rangle \in E_R(p); y \in E_C(C)\}| \leq n\} \\ & E_C(\texttt{restriction}(p, \texttt{qualifiedCardinality}(n, C))) \\ = \ & \{x \in O; |\{\langle x, y\rangle \in E_R(p); y \in E_C(C)\}| = n\} \end{aligned}$$

Consider the following expressions (in OWL 2):

```
ex:SmallTeam rdfs:subClassOf _:a .
_:a rdf:type owl:Restriction .
_:a owl:onProperty ex:member .
_:a owl:maxCardinality 5 .
ex:ModernTeam2 rdfs:subClassOf ex:SmallTeam .
ex:ModernTeam2 rdfs:subClassOf _:b .
_:b rdf:type owl:Restriction .
_:b owl:onProperty ex:member .
_:b owl:minQualifiedCardinality 4 .
_:b owl:onClass ex:Woman .
```

1. *Draw the graph corresponding to this set of triples.*

   See Figure 13.5.

2. *Express it in OWL/XML.*

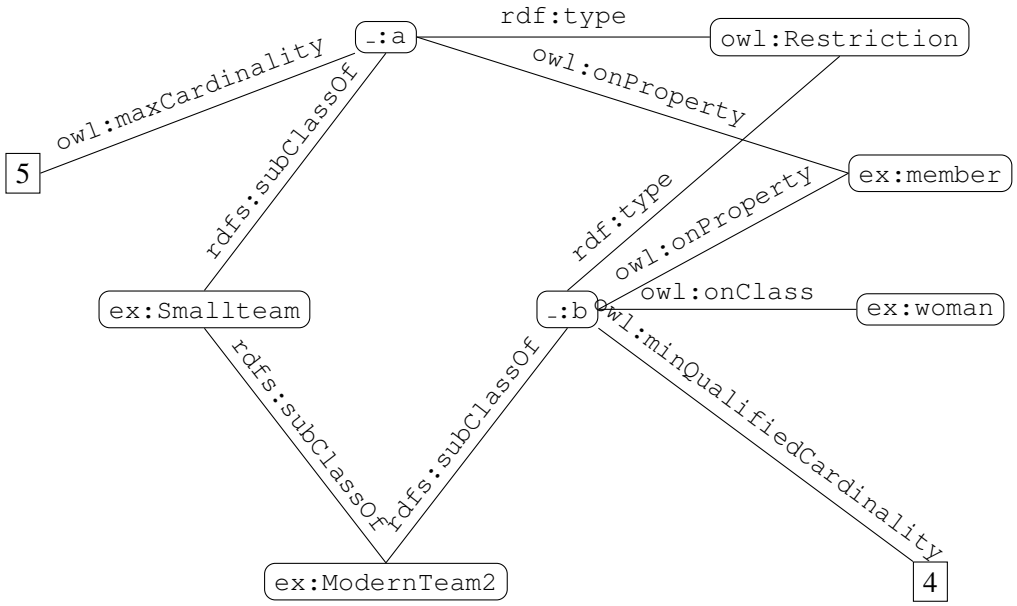


Figure 13.5: The RDF graph of the OWL 2 ontology.

```
<owl:Class rdf:about="#SmallTeam">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#member"/>
      <owl:maxCardinality
          rdf:datatype="&xsl;integer">5</owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:about="#ModernTeam2">
  <rdfs:subClassOf rdf:resource="#SmallTeam"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#member"/>
      <owl:minQualifiedCardinality
          rdf:datatype="&xsl;integer">4</owl:maxCardinality>
      <owl:onClass rdf:resource="#Woman"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

3. *Explain the meaning of this graph (paraphrase it in English)*

   "A modern team is a small team with at least 4 female team members; A small team is a team with at most 5 members."

4. *What would happen if we exchange the 5 and the 4?*

   Then the paraphrase would be that "A modern team is a small team with at least 5 female team members; A small team is a team with at most 4 members." hence, there could not exists any modern team.

Exercise 13 (From OWL 2 to OWL 1 and back)**.**

1. *How is it possible to rewrite* `qualifiedCardinality` *in function of the minimal and maximal qualified cardinality restrictions? Explain it with the semantics.*

   `qualifiedCardinality` *can be rewritten with respect to minimal and maximal qualified cardinality restrictions by replacing each of its occurences by the conjunction of the others:*

$$\begin{aligned}
&\texttt{restriction}(p, \texttt{qualifiedCardinality}(n, C)) \\
\equiv\ &\texttt{restriction}(p, \texttt{minQualifiedCardinality}(n, C)) \\
\sqcap\ &\texttt{restriction}(p, \texttt{maxQualifiedCardinality}(n, C))
\end{aligned}$$

   where $\sqcap$ *represent class conjunction (*`owl:intersectionOf`*). Indeed,*

$$\begin{aligned}
&E_C(\texttt{restriction}(p, \texttt{qualifiedCardinality}(n, C))) \\
=\ &\{x \in O; |\{\langle x, y\rangle \in E_R(p); y \in E_C(C)\}| = n\} \\
=\ &\{x \in O; n \leq |\{\langle x, y\rangle \in E_R(p); y \in E_C(C)\}| \leq n\} \\
=\ &\{x \in O; |\{\langle x, y\rangle \in E_R(p); y \in E_C(C)\}| \leq n \\
&\wedge |\{\langle x, y\rangle \in E_R(p); y \in E_C(C)\}| \geq n\} \\
=\ &\{x \in O; |\{\langle x, y\rangle \in E_R(p); y \in E_C(C)\}| \leq n\} \\
\cap\ &\{x \in O; |\{\langle x, y\rangle \in E_R(p); y \in E_C(C)\}| \geq n\} \\
=\ &E_C(\texttt{restriction}(p, \texttt{maxQualifiedCardinality}(n, C))) \\
\cap\ &E_C(\texttt{restriction}(p, \texttt{minQualifiedCardinality}(n, C))) \\
=\ &E_C(\texttt{restriction}(p, \texttt{maxQualifiedCardinality}(n, C)) \\
&\sqcap \texttt{restriction}(p, \texttt{minQualifiedCardinality}(n, C)))
\end{aligned}$$

2. *Is it possible to express* `minCardinality`*,* `maxCardinality`*,* `cardinality`*,* `someValuesFrom` *with these new qualified cardinality restrictions? Explain how.*

   It is possible:

$$\begin{aligned}
&\texttt{restriction}(p, \texttt{minCardinality}(n)) \\
\equiv\ &\texttt{restriction}(p, \texttt{minQualifiedCardinality}(n, \texttt{Thing})) \\
&\texttt{restriction}(p, \texttt{maxCardinality}(n)) \\
\equiv\ &\texttt{restriction}(p, \texttt{maxQualifiedCardinality}(n, \texttt{Thing})) \\
&\texttt{restriction}(p, \texttt{cardinality}(n)) \\
\equiv\ &\texttt{restriction}(p, \texttt{qualifiedCardinality}(n, \texttt{Thing})) \\
&\texttt{restriction}(p, \texttt{someValuesFrom}(C)) \\
\equiv\ &\texttt{restriction}(p, \texttt{minQualifiedCardinality}(1, C))
\end{aligned}$$

Consider, in addition to the previous RDF graphs, the following statements (expressed in OWL 1):

```
ex:womanmember owl:subPropertyOf ex:member .
ex:womanmember rdfs:range ex:Woman .
ex:ModernTeam1 rdfs:subClassOf ex:SmallTeam .
ex:ModernTeam1 rdfs:subClassOf _:b .
_:b rdf:type owl:Restriction .
_:b owl:onProperty ex:womanmember .
_:b owl:minCardinality 4 .
```

3. *Does* `ex:ModernTeam1` *subsume* `ex:ModernTeam2` *or the other way around? Justify.*

   It is clear that any `ex:ModernTeam1` *is a* `ex:ModernTeam2` *because the* `ex:womanmember` *are* `ex:member`*s who are* `ex:Woman`*, so having 4 of them will satisfy the required constraint in* `ex:ModernTeam2`*. Hence,* `ex:ModernTeam2` *subsumes* `ex:ModernTeam1`*.*

   The other way around is less clear a priori. However, when one interprets a statement like:

```
<owl:ObjectProperty rdf:about="#womanmember">
  <owl:subPropertyOf rdf:about="#member" />
  <rdfs:range rdf:about="#Woman" />
</owl:ObjectProperty>
```

   in OWL 1 semantics, this is interpreted definitionally, i.e., `ex:womanmember` *is equivalent to those* `ex:member` *who are* `ex:Woman`*. I.e.,*

$$
\begin{aligned}
& E_C(\texttt{restriction}(\texttt{ex:womanmember}, \texttt{minCardinality}(4))) \\
&= \{x \in O; |\{\langle x, y\rangle \in E_R(\texttt{ex:womanmember})| \geq 4\} \\
&= \{x \in O; |\{\langle x, y\rangle \in E_R(\texttt{ex:member} \sqcap \texttt{range}(\texttt{ex:Woman}))| \geq 4\} \\
&= \{x \in O; |\{\langle x, y\rangle \in E_R(\texttt{ex:member})\} \cap E_R(\texttt{range}(\texttt{ex:Woman}))| \geq 4\} \\
&= \{x \in O; |\{\langle x, y\rangle \in E_R(\texttt{ex:member})\} \cap \{\langle x, y\rangle; y \in E_C(\texttt{ex:Woman}))\}| \geq 4\} \\
&= \{x \in O; |\{\langle x, y\rangle \in E_R(\texttt{ex:member}); y \in E_C(\texttt{ex:Woman}))| \geq 4\} \\
&= E_C(\texttt{restriction}(\texttt{ex:member}, \texttt{minQualifiedCardinality}(4, \texttt{ex:Woman})))
\end{aligned}
$$

   Hence the two expressions are equivalent and `ex:ModernTeam1` *subsumes* `ex:ModernTeam2`*.*

4. *Does this suggest that it is also possible to express qualified cardinality constraints in OWL 1? Explain.*

   This indeed suggests that, by adding property definitions, such as `ex:womanmember`*, it is possible to transform any OWL 1 ontology with qualified cardinality restrictions into an equivalent plain OWL 1 ontology, i.e., without qualified cardinality restrictions.*

5. *Does qualified cardinality restrictions provide additional expressivity to OWL 1?*

   Hence, no. Qualified cardinality restrictions are only syntactic sugar.

Exercise 14 (FRBR in RDFS)**.** *FRBR is now a well established vocabulary in libraries. FRBR distinguishes between:*

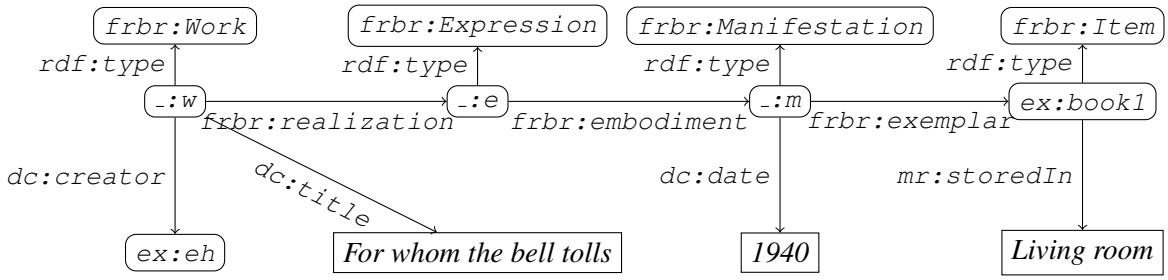
- *a* work *which is an abstract idea of what a work is;*
- *an* expression *which is a realization of this work in a particular form (poetry, music, painting);*
- *a* manifestation *which is a distinct embodiment of the expression (an edition of a text, a release of a movie; the reproduction of a painting);*
- *an* item *which is an often physical exemplar of a manifestation (an exemplar of a book; a copy of an MP3 file).*

Consider a fragment S of its RDF Schema manifestation:

```
frbr:Work rdf:type rdfs:Class .
frbr:Expression rdf:type rdfs:Class .
frbr:Manifestation rdf:type rdfs:Class .
frbr:Item rdf:type rdfs:Class .
frbr:Performance rdfs:subClassOf frbr:Expression .
```

```
frbr:adaptation rdfs:domain frbr:Work .
frbr:adaptation rdfs:range frbr:Work .
frbr:realization rdfs:domain frbr:Work .
frbr:realization rdfs:range frbr:Expression .
frbr:translation rdfs:domain frbr:Expression .
frbr:translation rdfs:range frbr:Expression .
frbr:embodiment rdfs:domain frbr:Expression .
frbr:embodiment rdfs:range frbr:Manifestation .
frbr:exemplar rdfs:domain frbr:Manifestation .
frbr:exemplar rdfs:range frbr:Item .
```

For instance, the first 5 statements of the graph G of Exercise 4 in FRBR would correspond to the following graph:



1. *How would you extend the above S with the vocabulary identified by the prefix* `mr` *in graph G of Exercise 4? I.e., give triples using the RDFS vocabulary (*`rdfs:subClassOf`*,* `rdfs:subPropertyOf`*,* `rdfs:domain`*,* `rdfs:range`*) to extend FRBR with the corresponding entities.*

   ```
   mr:Book rdfs:subClassOf frbr:Manifestation .
   mr:Movie rdfs:subClassOf frbr:Manifestation .
   mr:storedIn rdfs:domain frbr:Item .
   mr:storedIn rdfs:range xsd:string .
   mr:translationOf rdfs:subPropertyOf frbr:translation .
   mr:translationOf rdfs:domain frbr:Expression .
   mr:translationOf rdfs:range frbr:Expression .
   mr:adaptedFrom rdfs:subPropertyOf frbr:adaptation .
   mr:adaptedFrom rdfs:domain frbr:Work .
   mr:adaptedFrom rdfs:range frbr:Work .
   ```

Exercise 15 (RDFS Entailment)**.** *Given* P'=$\{$ *?x* `rdf:type` `a:Hospital` *. ?x* `a:hasService` *?y . ?y* `rdf:type` `a:Maternity` *. ?x* `a:hasService` *?z . ?z* `rdf:type` `a:Pediatry` *.*$\}$ *and G the graph of Figure 2.8 (p. 29),*

1. *Does* $G \models_{RDF} P'$*? (explain how and/or why)*

   No. Because there is only one `a:Hospital` *in G, but there is no triple in G stating that it has a service whose* `rdf:type` *is* `a:Maternity`*.*

2. *Which procedure may be used to derive that* $G \models_{RDFS} P'$*?*

   The relevant procedure would be to compute the partial ter Horst closure $\hat{G}\backslash P'$ *and show that* $\hat{G}\backslash P' \models P'$*.*

Exercise 16 (RDFS and OWL interpretation)**.**

1. *One convenient way to interpret together two heterogeneous sources is to interpret them through a common ontology. Consider the ontology O made of the following statements:*

```
o:parent rdfs:domain foaf:Person .
o:parent rdfs:range foaf:Person .
o1:mother rdfs:subPropertyOf o:parent .
o1:father rdfs:subPropertyOf o:parent .
o1:mother rdfs:domain o2:Female .
o1:father rdfs:domain o2:Male .
```

   Considering graphs G_1, G_2, and Q_1 of Exercise 6 (p.28), does $O \cup G_1 \cup G_2 \models_{RDF} Q_1$?

   No, because it would still need to contain `o:parent` statements which are not in any of O, G_1 or G_2 (in O, `o:parent` is used as subject and object but not as predicate).

2. *Does $O \cup G_1 \cup G_2 \models_{RDFS} Q_1$? (explain your answer)*

   Yes, because through the ter Horst closure it is possible to saturate the graph $O \cup G_1 \cup G_2$ and, in particular, each time `o1:father` and `o1:mother` are used, the same triple can be added with `o:parent` as predicate, due to rule [RDFS 9]. So the graph made of:

```
a:Stephanie o1:mother a:Sylvie .          a:Stephanie o1:mother a:William .
a:Sylvie o2:attendsClass b:5e2 .          a:William o2:attendsClass b:5e2 .
a:Sylvie rdf:type o2:Female .             a:William rdf:type o2:Male .
```

   is a subgraph of the closure of $O \cup G_1 \cup G_2$. It is also an instance of the graph Q_1 through the assignment: $\{$`?x` $\leftarrow$ `a:William`, `?y` $\leftarrow$ `a:Sylvie`, `?w` $\leftarrow$ `b:5e2`, `?z` $\leftarrow$ `a:Stephanie`$\}$

   Give all mappings (variable/blank assignments) which support this entailment.

?x	?y	?w	?z
a:William	a:Sylvie	b:5e2	a:Stephanie
a:William	a:Sylvie	b:5e2	_:b1

   What additional facts does $O \cup G_1 \cup G_2$ RDFS-entail? (provide an example).

   It allows to deduce the gender of parents (through the `owl:domain` constraints on the `o1:father` and `o1:mother` properties). So, for instance, `a:Pierre rdf:type o2:Male` and `a:Marie rdf:type o2:Female`.

3. *Can you express in OWL the class `o:ParentOfNumerousChildren`, as the class of those parents with more than three children, using the concepts and properties of ontology O?*

$$\texttt{o:ParentOfNumerousChildren} \sqsubseteq \geq_3 \texttt{o:parent}$$

   or

```
<owl:Class rdf:about="o:ParentOfNumerousChildren">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="o:parent" />
      <owl:minCardinality>3</owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

Give the interpretation of this (compound) class.

$$I(\texttt{o:ParentOfNumerousChildren}) \subseteq I(\geq_3 \texttt{o:parent})$$
$$= \{o \in O \mid ||\{\langle o, y\rangle \in I(\texttt{o:parent})\}|| \geq 3\}$$

Exercise 17 (RDFS and OWL entailment)**.** *Consider the ontology O made of the following statements:*

```
o:accomodation rdfs:range o:Accomodation .
o:Local rdfs:subClassOf o:Accomodation .
o:Pousada rdfs:subClassOf o:Local .
o:Ryokan rdfs:subClassOf o:Local .
o:GrandHotel rdfs:subClassOf Accomodation .

o:activity rdfs:range o:Activity .
o:Sport rdfs:subClassOf o:Activity .
o:Swimming rdfs:subClassOf o:Sport .
o:SwordFighting rdfs:subClassOf o:Sport .
o:Visit rdfs:subClassOf o:Activity .
o:Cruising rdfs:subClassOf o:Visit .
```

and the graph G of Exercise 7

1. *Does $G \models_{RDFS}$ `o:Package rdf:type rdfs:Class`?*
   Does $O \models_{RDFS}$ `o:Package rdf:type rdfs:Class`?

   $G \models_{RDFS}$ `o:Package rdf:type rdfs:Class`

   Because `o:Package` is the `rdf:type` of items, this entails that it is a class. Indeed, by the RDF semantics (1), $\langle \iota'(\texttt{\_:b1}), \iota(\texttt{o:Pachage})\rangle \in I_{EXT}(\iota(\texttt{rdf:type}))$; but all axiomatic triples are satisfied (2c) and in particular $\langle$`rdf:type rdfs:range rdfs:Class`$\rangle$ which means that (6c), $\iota(\texttt{o:Package}) \in I_{CEXT}(\iota(\texttt{rdfs:Class}))$ and (6c) $\langle \iota(\texttt{o:Package}), \iota(\texttt{rdfs:Class})\rangle \in I_{EXT}(\iota(\texttt{rdf:type}))$. Since, this is true for all models of G, this means that all these models satisfy $\langle$`o:Package rdf:type rdfs:Class`$\rangle$.

   $O \not\models_{RDFS}$ `o:Package rdf:type rdfs:Class`

   Because, since there is no mention of `o:Package` in O, this does not allow to entail anything about it. More precisely, there is no constraint in O preventing that $\iota(\texttt{o:Package}) \in I_R \setminus Class$.

2. *Does $O \cup G \models_{RDF} K$? $O \cup G \models_{RDFS} K$? Explain why.*

   $O \cup G \not\models_{RDF} K$

   For this to be satisfied, it would be necessary that an instance of K be a subgraph of $O \cup G$. This would necessitate a triple whose predicate is `rdf:type` and whose object be `o:Sport`. But no such triple exist either in O or in G.

   $O \cup G \models_{RDFS} K$

   Indeed, if one computes the (partial) closure of $O \cup G$, then it contains $\langle$`_:b5`, `rdf:type`, `o:Sport`$\rangle$ (and $\langle$`d:ToyofukuRyokan`, `rdf:type`, `o:Local`$\rangle$) by rule [RDFS11] because, G contains $\langle$`_:b5`, `rdf:type`, `o:SwordFighting`$\rangle$ (and $\langle$`d:ToyofukuRyokan`, `rdf:type`, `o:Ryokan`$\rangle$) and O contains $\langle$`o:Ryokan`, `rdfs:subClassOf`, `o:Local`$\rangle$ (and $\langle$`o:SwordFighting`, `rdfs:subClassOf`, `o:Sport`$\rangle$). Thus, it is possible to define an RDF-homomorphism $h : K \to cl(O \cup G)$ such that $h($`_:y`$) =$ `_:b4`, $h($`_:acc`$) =$ `d:ToyofukuRyokan`, $h($`_:act`$) =$ `_:b5` and $h(K) \in cl(O \cup G)$. h is indeed an homomorphism as it preserves the graph structure of K.

3. *Given the OWL axiom (making the OWL ontology O'):*

$$\begin{aligned}\texttt{o:TonicPackage} \equiv\ & \texttt{o:Package} \\ & \sqcap \exists \texttt{o:accomodation}.(\texttt{o:Local} \sqcap \geq_1 \texttt{o:swimmingPool}) \\ & \sqcap \exists \texttt{o:activity}.\texttt{o:Sport}\end{aligned}$$

Give the OWL interpretation of TonicPackage ($E_C(\texttt{o:TonicPackage})$).

$$\begin{aligned}E_C(\texttt{o:TonicPackage}) =\ & E_C(\texttt{o:Package} \\ & \quad \sqcap \exists \texttt{o:accomodation}.(\texttt{o:Local} \sqcap \geq_1 \texttt{o:swimmingPool}) \\ & \quad \sqcap \exists \texttt{o:activity}.\texttt{o:Sport}) \\ =\ & E_C(\texttt{o:Package}) \\ & \cap E_C(\exists \texttt{o:accomodation}.(\texttt{o:Local} \sqcap \geq_1 \texttt{o:swimmingPool})) \\ & \cap E_C(\exists \texttt{o:activity}.\texttt{o:Sport})) \\ =\ & E_C(\texttt{o:Package}) \\ & \cap \{x | \langle x, y\rangle \in E_R(\texttt{o:accomodation}) \\ & \quad \wedge y \in E_C(\texttt{o:Local} \sqcap \geq_1 \texttt{o:swimmingPool})\} \\ & \cap \{x | \langle x, y\rangle \in E_R(\texttt{o:activity}) \wedge y \in E_C(\texttt{o:Sport})\} \\ =\ & E_C(\texttt{o:Package}) \\ & \cap \{x | \langle x, y\rangle \in E_R(\texttt{o:accomodation}) \\ & \quad \wedge y \in E_C(\texttt{o:Local}) \cap E_C(\geq_1 \texttt{o:swimmingPool})\} \\ & \cap \{x | \langle x, y\rangle \in E_R(\texttt{o:activity}) \wedge y \in E_C(\texttt{o:Sport})\} \\ =\ & E_C(\texttt{o:Package}) \\ & \cap \{x | \langle x, y\rangle \in E_R(\texttt{o:accomodation}) \\ & \quad \wedge y \in E_C(\texttt{o:Local}) \\ & \quad\quad \cap \{z | \#\{\langle z, t\rangle \in E_R(\texttt{o:swimmingPool})\} \geq 1\}\} \\ & \cap \{x | \langle x, y\rangle \in E_R(\texttt{o:activity}) \wedge y \in E_C(\texttt{o:Sport})\}\end{aligned}$$

4. *Does $O \cup O' \cup G \models_{OWL}$ `_:b1 rdf:type o:TonicPackage`? Tell why.*

The definition of `o:TonicPackage` constraints its instances have an accomodation that has at least one swimming pool. However, neither O nor G refer to `o:swimmingPool`, hence there can be models of $O \cup O' \cup G$ in which $E_C(\texttt{o:swimmingPool}) = \emptyset$ and thus $E_C(\texttt{o:TonicPackage}) = \emptyset$. Obviously, such models do not satisfy `_:b1 rdf:type o:TonicPackage`. Hence this statement is not a consequence.

SPARQL

Exercise 18 (SPARQL queries)**.** *Given the following SPARQL query:*

```
SELECT ?t
PREFIX
  foaf: http://xmlns.com/foaf/0.1/
  m: http://mydomain.com/myExample#
  dc: http://purl.org/dc/elements/1.1/
```

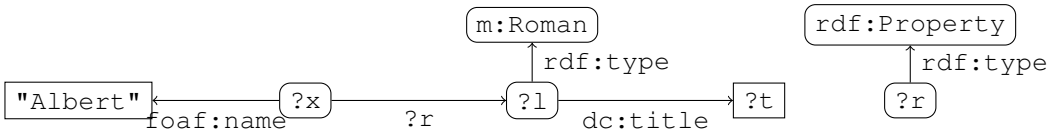


Figure 13.6: SPARQL graph patterns.

```
  rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#
WHERE {
  ?x foaf:name "Albert".
  ?x ?r ?l.
  ?r rdf:type rdf:Property.
  ?l rdf:type m:Roman.
  ?l dc:title ?t.
}
```

1. *What is the informal meaning of this query?*

   Find all titles (`dc:title`) of novels (`m:Roman`) related (`?r`) to an entity (`?x`) whose name (`foaf:name`) is "Albert".

2. *Draw the RDF graph corresponding to the graph patterns of the query.*

   It is given in Figure 13.6.

3. *What is the difference between such graph patterns and simple RDF graphs?*

   These graph patterns are generalised RDF graphs*: they can contain blanks or variables as edge labels (or as properties in RDF triple terms).*

4. *Evaluate this query on each of the well-founded graphs of Figure 2.6 of Exercise 1 (p26) and provide the answer.*

   (a) : $\{\langle$ `"La peste"` $\rangle\}$
   (c) : $\{\}$ *(`?l` is a book (`Livre`) but not a novel (`Roman`))*
   (d) : $\{\langle$ `"La peste"` $\rangle\}$

5. *What must be added to this query for returning the year of publication of the `m:Roman` **if it is available**?*

   An optional clause:

   ```
   SELECT ?t ?y
   ...
   OPTIONAL { ?l m:annee ?y. }
   ```

Exercise 19 (Simple SPARQL query)**.**

1. *Can you express a SPARQL query returning all `o:TonicPackage` as defined in the OWL axiom of question 3 of Exercise 17?*

   ```
   SELECT ?p
   WHERE {
   ```

```
        ?p rdf:type o:Package .
        ?p o:accomodation ?acc .
        ?acc rdf:type o:Local .
        ?acc o:swimmingPool ?sw .
        ?p o:activity ?act .
        ?act rdf:type o:Sport .
    }
```

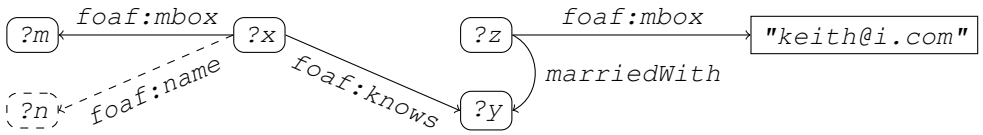
Exercise 20 (SPARQL and queries)**.** *Given the following SPARQL query:*

```
SELECT ?m, ?n
PREFIX  foaf: http://xmlns.com/foaf/0.1/
WHERE {
  ?x foaf:mbox ?m.
  ?x foaf:knows ?y.
  ?z foaf:mbox "keith@i.com".
  ?z marriedWith ?y.
  OPTIONAL { ?x foaf:name ?n }
}
```

1. *What is, informally, the meaning of this query (tell it in English or French)?*

   What are the email address, and if available the name, of people who know the spouse of the person with email address "keith@i.com"?

2. *Draw the GRDF graph corresponding to the graph pattern of this query.*



3. *Evaluate the query on the graphs of Figure 2.7 of Exercise 2 (p27) and provide the results.*

   For both graphs, there is no result because there is no node which is both the object of a `foaf:knows` *and a* `marriedWith` *(for matching* `?y`*).*

4. *Evaluate this query on the closure obtained at Question 4 of Exercise 9 and provide the results.*

   (a) still no result because this time there is no node whose `foaf:mbox` *is* `"keith@i.com"` *which is the subject of a* `marriedWith` *relation, (b) 1 result:* $\{\langle$ `?m`, `"paul@e.at"` $\rangle, \langle$ `?n`, `null` $\rangle\}$ *because,* $\langle$ `?a`, `foaf:knows`, `?c` $\rangle$ *belongs to the closure of (b).*

Exercise 21 (SPARQL query containment)**.** *Consider the following queries* q_1 *and* q_2 *on the RDF graph* G *of Exercise 3:*

$q_1 =$ `SELECT` $?w$ `FROM` G `WHERE` $(\langle ?w$ `o:wrote` $?x\rangle$ `AND` $\langle ?x$ `rdf:type o:Poem`$\rangle)$ `UNION` $\langle ?w$ `o:translated` $?x\rangle$

$q_2 =$ `SELECT` $?w$ `FROM` G `WHERE` $(\langle ?w$ `o:wrote` $?l\rangle$ `UNION` $\langle ?w$ `o:translated` $?l\rangle)$ `AND` $\langle ?l$ `rdf:type o:Poem`$\rangle$

1. *In the course, we defined the distinguished variables $\vec{B}$, the queried graph G and the query pattern P. Identify them in q_1 and q_2.*

   In both cases, $\vec{B}$ is $\langle ?w \rangle$ and G is G. The patterns of q_1 and q_2 are respectively :

   $$P_1 = (\langle ?w \texttt{ o:wrote } ?x \rangle \texttt{ AND } \langle ?x \texttt{ rdf:type o:Poem} \rangle) \texttt{ UNION } \langle ?w \texttt{ o:translated } ?x \rangle$$
   $$P_2 = (\langle ?w \texttt{ o:wrote } ?l \rangle \texttt{ UNION } \langle ?w \texttt{ o:translated } ?l \rangle) \texttt{ AND } \langle ?l \texttt{ rdf:type o:Poem} \rangle$$

2. *Provide the answers of q_1 and q_2 with respect to the graph G.*

   The answers to query q_1 and q_2 on the graph G are respectively $\{\langle \texttt{d:Poe} \rangle, \langle \texttt{d:Mallarmé} \rangle, \langle \texttt{d:Baudelaire} \rangle\}$ and $\{\langle \texttt{d:Poe} \rangle, \langle \texttt{d:Mallarmé} \rangle\}$.

Query containment $q \sqsubseteq q'$ between two queries $q =$ SELECT $\vec{B}$ FROM G WHERE P and $q' =$ SELECT $\vec{B}$ FROM G WHERE P' is defined by the fact that for any RDF graph, the answers to q are included in those to q' ($\forall G, \mathcal{A}(\vec{B}, G, P) \subseteq \mathcal{A}(\vec{B}, G, P')$).

3. *What does the answer to the previous questions tell you about query containment between q_1 and q_2?*

   $q_1 \not\sqsubseteq q_2$ because G is a counter example: $\{\langle \texttt{d:Poe} \rangle, \langle \texttt{d:Mallarmé} \rangle, \langle \texttt{d:Baudelaire} \rangle\} = \mathcal{A}(\langle ?w \rangle, G, P_1) \not\subseteq \mathcal{A}(\langle ?w \rangle, G, P_2) = \{\langle \texttt{d:Poe} \rangle, \langle \texttt{d:Mallarmé} \rangle\}$.

4. *Do you think that query containment holds in some direction between q_1 and q_2 (either $q_1 \sqsubseteq q_2$ or $q_2 \sqsubseteq q_1$)?*

   $q_2 \sqsubseteq q_1$.

5. *Provide a proof for this. This may be done semantically by using the interpretation of query patterns or syntactically by translating queries into logic and showing that the query containment statement is a theorem.*

   The argument for the proof is that $P_1 = (A \wedge B) \vee C$ and $P_2 = (A \vee C) \wedge B$, but $(A \vee C) \wedge B = (A \wedge B) \vee (C \wedge B)$. Hence, P_2 is more specific than P_1. More formally:

   $$\begin{aligned}
   & \sigma \in A(\langle ?w \rangle, G, P_2) \\
   \Leftrightarrow\ & G \models \sigma((\langle ?w \texttt{ o:wrote } ?l \rangle \texttt{ UNION } \langle ?w \texttt{ o:translated } ?l \rangle) \\
   & \qquad \texttt{AND } \langle ?l \texttt{ rdf:type o:Poem} \rangle) \\
   \Leftrightarrow\ & G \models \sigma(\langle ?w \texttt{ o:wrote } ?l \rangle \texttt{ UNION } \langle ?w \texttt{ o:translated } ?l \rangle) \\
   & \text{and } G \models \sigma(\langle ?l \texttt{ rdf:type o:Poem} \rangle) \\
   \Leftrightarrow\ & (G \models \sigma(\langle ?w \texttt{ o:wrote } ?l \rangle) \text{ or } G \models \sigma(\langle ?w \texttt{ o:translated } ?l \rangle)) \\
   & \text{and } G \models \sigma(\langle ?l \texttt{ rdf:type o:Poem} \rangle) \\
   \Leftrightarrow\ & (G \models \sigma(\langle ?w \texttt{ o:wrote } ?l \rangle) \text{ and } G \models \sigma(\langle ?l \texttt{ rdf:type o:Poem} \rangle)) \\
   & \quad \text{or } (G \models \sigma(\langle ?w \texttt{ o:translated } ?l \rangle) \text{ and } G \models \sigma(\langle ?l \texttt{ rdf:type o:Poem} \rangle)) \\
   \Rightarrow\ & (G \models \sigma(\langle ?w \texttt{ o:wrote } ?l \rangle) \text{ and } G \models \sigma(\langle ?l \texttt{ rdf:type o:Poem} \rangle)) \\
   & \text{or } G \models \sigma(\langle ?w \texttt{ o:translated } ?l \rangle) \\
   \Leftrightarrow\ & G \models \sigma(\langle ?w \texttt{ o:wrote } ?x \rangle \texttt{ AND } \langle ?x \texttt{ rdf:type o:Poem} \rangle) \\
   & \text{or } G \models \sigma(\langle ?w \texttt{ o:translated } ?x \rangle) \\
   \Leftrightarrow\ & G \models \sigma((\langle ?w \texttt{ o:wrote } ?x \rangle \texttt{ AND } \langle ?x \texttt{ rdf:type o:Poem} \rangle) \\
   & \qquad \texttt{UNION } \langle ?w \texttt{ o:translated } ?x \rangle) \\
   \Leftrightarrow\ & \sigma \in A(\langle ?w \rangle, G, P_1)
   \end{aligned}$$

   Hence, $q_2 \sqsubseteq q_1$.

Query modulo ontologies

Exercise 22 (SPARQL modulo ontologies)**.**

1. *The way queries are answered in Exercise 20 is not the standard way for answering queries modulo DL-Lite ontologies. What is the standard method? Rewrite the above query with this method and give its results.*

   The usual way to evaluate queries modulo DL-Lite is to rewrite the query as the following:

```
SELECT ?m, ?n
PREFIX  foaf: http://xmlns.com/foaf/0.1/
WHERE {
  ?x foaf:mbox ?m.
  { { ?x foaf:knows ?y. }
    UNION { ?x foaf:playsSportWith ?y. }
    UNION  { ?x foaf:playsTennisWith ?y. }
    UNION { ?x foaf:worksWith ?y. }
    UNION  { ?x marriedWith ?y.}
  }
  ?z foaf:mbox "keith@i.com".
  ?z marriedWith ?y.
  OPTIONAL { ?x foaf:name ?n }
}
```

   It can be checked that this query will return the same results as in Question 4 of Exercise 20.

Exercise 23 (Query modulo ontology)**.** *We now consider the ontology o of Exercise 11 and the following queries:*

- $q_3 =$ `SELECT` $?y$ `FROM` o `WHERE` $\langle ?x, \texttt{o:translated}, ?y\rangle$;
- $q_4 =$ `SELECT` $?y$ `FROM` o `WHERE` $\langle ?y, \texttt{rdf:type}, \texttt{o:Literature}\rangle$.

1. *Do you think that query containment holds in some direction between q_3 and q_4 (either $q_3 \sqsubseteq q_4$ or $q_4 \sqsubseteq q_3$)? Tell why.*

   None of these because SPARQL evaluates queries by finding triples in the graph and the triples are not comparable. More formally, assume $G_1 = \{\langle a, \texttt{o:translated}, b\rangle\}$ and $G_2 = \{\langle c, \texttt{rdf:type}, \texttt{o:Literature}\rangle\}$, it is clear that $\mathcal{A}(q_3, G_1) \not\subseteq \mathcal{A}(q_4, G_1)$ and $\mathcal{A}(q_4, G_2) \not\subseteq \mathcal{A}(q_3, G_2)$. Hence, there cannot be any containment between these queries.

2. *Can you provide a definition for query containment modulo an ontology o ($q \sqsubseteq_o q'$)?*

   There is no reason to change the structure of the definition: Query containment $q \sqsubseteq q'$ between two queries $q =$ `SELECT` $\vec{B}$ `FROM` o `WHERE` P and $q' =$ `SELECT` $\vec{B}$ `FROM` o `WHERE` P' is defined by the fact that for any RDFS ontologies, the answers to q are included in those to q' ($\forall o, \mathcal{A}^+(\vec{B}, o, P) \subseteq \mathcal{A}^+(\vec{B}, o, P')$).

   Everything is in the definition of $\mathcal{A}^+$. A natural semantic definition would be that:

$$\mathcal{A}^+(\vec{B}, o, P) = \{\sigma|^{\vec{B}}_{\vec{B}} | o \models_{RDFS} \sigma(P)\}$$

   or a more pragmatic approach would be to define it with the closure:

$$\mathcal{A}^+(\vec{B}, o, P) = \mathcal{A}(\vec{B}, \hat{o}\backslash\backslash P, P)$$

3. *Does it return different answers for q_3 and q_4?(either $q_3 \sqsubseteq_o q_4$ or $q_4 \sqsubseteq_o q_3$)? Tell why.*

   From the definition of o, it is clear that whenever $o \models_{RDFS} \langle a, \texttt{o:translated}, b\rangle$, then $o \models_{RDFS} \langle b, \texttt{rdf:type}, \texttt{o:Literature}\rangle$. The converse is not true (there exists models satisfying $\langle b, \texttt{rdf:type}, \texttt{o:Literature}\rangle$ but no a such that $\langle a, \texttt{o:translated}, b\rangle$, i.e., there may be non translated books). Then for any $\langle b\rangle \in \mathcal{A}^+(q_3)$, $\langle b\rangle \in \mathcal{A}^+(q_4)$, so $q_3 \sqsubseteq_o q_4$. But not the other way around.

Exercise 24 (SPARQL queries)**.** *Consider the following statements added to the statements of Exercise 12 (p.55):*

```
ex:MyTeam rdf:type ex:ModernTeam2 .
ex:Kay ex:member ex:MyTeam .
ex:Kay rdf:type ex:Man .
ex:Jo ex:member ex:MyTeam .
```

1. *If one queries this graph with* `SELECT ?x WHERE ?x ex:member ex:MyTeam . ?x rdf:type ex:Woman .`*, what would be the answer?*

   The answer would be empty because no team member is explicitely declared as a `ex:Woman`*. Hence there is no way to project the triple pattern* `?x rdf:type ex:Woman .` *into a triple of the graph.*

   Even if queries were interpreted with respect to ontologies, this would not be sufficient to entail that `ex:Jo` *is a* `ex:Woman` *because it is not excluded that* `ex:Jo` *and* `ex:Kay` *are different identifiers for the same resource. Even then, it is not prevented by the ontology that* `ex:Woman` *and* `ex:Man` *are disjoint, hence* `ex:Kay` *may be both and* `ex:Jo` *not be a* `ex:Woman`*.*

2. *What would be necessary for* `ex:Jo` *to be an answer?*

   The simplest solution would be to have `ex:Jo rdf:type ex:Woman .` *in the graph.*

   If queries are interpreted modulo ontologies, it would be sufficient to have:

   ```
   ex:Man owl:disjointWith ex:Woman .
   ex:Jo owl:differentFrom ex:Kay .
   ```

   in the graph.

Exercise 25 (Refactoring graphs with SPARQL CONSTRUCT)**.**

1. *Both movies and books may be considered as work expressions. Write a SPARQL query able to return all such items in graph G of Exercise 4 with their title and, if possible, the place they are stored in.*

   ```
   SELECT ?item ?title ?place
   PREFIX ...
   FROM G
   WHERE {
     ?item dc:title ?title .
     { ?item rdf:type mr:Movie } UNION { ?item rdf:type mr:Book }
   } OPTIONAL { ?item mr:storedIn ?place }
   ```

2. *Consider that, instead of extending the schema, one would prefer to refactor the graph G of Exercise 4 so that it corresponds to the schema S of Exercise 14. Create a SPARQL CONSTRUCT query able to extract the data from G and generate a graph complying to S (check on the example above).*

```
CONSTRUCT {
  _:w rdf:type frbr:Work .
  _:w dc:creator ?creator .
  _:w dc:title ?title .
  _:w frbr:realization _:e .
  _:e rdf:type frbr:Expression .
  _:e frbr:embodiement _:m .
  _:m rdf;type frbr:Manifestation .
  _:m dc:date ?date .
  _:m frbr:exemplar ?item .
  ?item rdf:type frbr:Item .
  ?item mr:storedIn ?place .
}
PREFIX ...
FROM G
WHERE {
  ?item rdf:type mr:Book .
  ?item dc:title ?title .
  ?item dc:date ?date .
  ?item dc:creator ?creator .
  ?item mr:storedIn ?place .
}
```

3. *Considering that you have a SPARQL engine able to answer queries taking into account RDFS semantics, write the same query as in Question 1 using the schema S and the answer to Question 1.*

```
SELECT ?item ?title ?place
PREFIX ...
FROM G
WHERE {
  ?y dc:title ?title .
  ?y frbr:realization ?x .
  ?x frbr:embodiement ?e .
  ?e frbr:exemplar ?item .
} OPTIONAL { ?item mr:storedIn ?place }
```

Exercise 26 (Computing closure with SPARQL CONSTRUCT)**.** *It is possible to generate RDF graphs from an RDF graph through the use of SPARQL CONSTRUCT. Consider the query Q:*

CONSTRUCT { *?y* `rdf:type` *?x* }
WHERE { *?u* `rdfs:subClassOf` *?x* . *?y* `rdf:type` *?u* }.

1. *Apply Q to the graph G of Figure 2.8 (p. 29) and give the resulting graph. Let us call the result $Q(G)$, does $G \cup Q(G) \models_{RDF} P'$?*

```
b:H.Michallon rdf:type a:Hospital .
b:GO rdf:type a:Maternity .
```

$G \cup Q(G) \models_{RDF} P'$ because by substituting in P' ?x by `b:Belledonne`*, ?y by* `b:GO` *and ?z by* `b:Pediatry`*, we obtain a subgraph of $G \cup Q(G)$.*

2. *For any map σ, does*

   $$\sigma(?u)\ \texttt{rdfs:subClassOf}\ \sigma(?x).\ \sigma(?y)\ \texttt{rdf:type}\ \sigma(?u) \models_{RDFS} \sigma(?y)\ \texttt{rdf:type}\ \sigma(?x)$$

   (explain why)

   Yes, because for any RDFS model $\langle I_R, I_P, Class, I_{EXT}, I_{CEXT}, Lit, \iota\rangle$ of the premises, $\iota(\sigma(?y)) \in I_{CEXT}(\iota(\sigma(?u)))$ and $I_{CEXT}(\iota(\sigma(?u))) \subseteq I_{CEXT}(\iota(\sigma(?x)))$, hence $\iota(\sigma(?y)) \in I_{CEXT}(\iota(\sigma(?x)))$, which means that $\sigma(?y)$ `rdf:type` $\sigma(?x)$ is satisfied by any model of the premises.

3. *Is this related with the rules of the ter Horst closure? (tell if it corresponds more closely to one of these rules).*

   In fact, this query is a transcription of [RDFS11].

4. *Would it be possible to transform all closure rules as SPARQL CONSTRUCTS? Do it or explain why.*

   Rules [RDF1] and [RDFS1] could either be rendered by a CONSTRUCT with empty WHERE part or simply added to the graph G. Rule [RDF3] is a bit difficult to transcribe the WHERE clause. Otherwise all other rules can be expressed as SPARQL CONSTRUCT as follows:

[Q2]	*CONSTRUCT*	{ *?p* `type` `prop` }	*WHERE* { *?s ?p ?o* }
[Q6]	*CONSTRUCT*	{ *?u* `type` *?x* }	*WHERE* { *?u ?a ?v* . *?a* `dom` *?x* }
[Q7]	*CONSTRUCT*	{ *?v* `type` *?x* }	*WHERE* { *?u ?a ?v* . *?a* `range` *?x* }
[Q8a]	*CONSTRUCT*	{ *?x* `sp` *?x* }	*WHERE* { *?x* `type` `prop` }
[Q8b]	*CONSTRUCT*	{ *?x* `sp` *?z* }	*WHERE* { *?x* `sp` *?y* . *?y* `sp` *?z* }
[Q9]	*CONSTRUCT*	{ *?x ?b ?y* }	*WHERE* { *?x ?a ?y* . *?a* `sp` *?b* }
[Q10]	*CONSTRUCT*	{ *?x* `sc` `res` }	*WHERE* { *?x* `type` `class` }
[Q11]	*CONSTRUCT*	{ *?y* `type` *?x* }	*WHERE* { *?u* `sc` *?x* . *?y* `type` *?u* }
[Q12a]	*CONSTRUCT*	{ *?x* `sc` *?x* }	*WHERE* { *?x* `type` `class` }
[Q12b]	*CONSTRUCT*	{ *?x* `sc` *?z* }	*WHERE* { *?x* `sc` *?y* . *?y* `sc` *?z* }
[Q13]	*CONSTRUCT*	{ *?x* `sp` `member` }	*WHERE* { *?x* `type` `contMP` }
[Q14]	*CONSTRUCT*	{ *?x* `sc` `literal` }	*WHERE* { *?x* `type` `datatype` }

5. *What more is needed for computing the closure?*

   SPARQL queries are "one shot", for computing the closure it is necessary that the result of rules be applied recursively until they do not generate new triples.

   (Q2,Q6,Q7,Q8a,Q8b,Q9,Q10,Q11,Q12a,Q12b,Q13,Q14)($RDF1 \cup RDFS1 \cup G$)*

Network of ontologies

Exercise 27 (Network of ontologies)**.** *In addition of the ontology o of Exercise 11, we consider an ontology o' which defines the class* `op:Buch` *and contains the following statements:*

$$\langle \texttt{d:Baudelaire}, \texttt{o:translated}, \texttt{d:Confessions}\rangle\ \langle \texttt{d:DeQuincey}, \texttt{o:wrote}, \texttt{d:Confessions}\rangle$$

and o'' which defines the class `opp:Roman` *and contain the following statements:*

$$\langle \texttt{d:Confessions}, \texttt{rdf:type}, \texttt{opp:Roman}\rangle\ \langle \texttt{d:Musset}, \texttt{o:translated}, \texttt{d:Confessions}\rangle$$

They are related together by the following three alignments:

– $A_{o,o'} = \{\langle \texttt{o:Literature}, \equiv, \texttt{op:Buch}\rangle\}$

- $A_{o',o''} = \{\langle \texttt{op:Buch}, \sqsubseteq, \texttt{opp:Roman}\rangle\}$
- $A_{o'',o} = \{\langle \texttt{opp:Roman}, \equiv, \texttt{o:Novel}\rangle\}$

So that we have a network of ontology $\langle\{o, o', o''\}, \{A_{o,o'}, A_{o',o''}, A_{o'',o}\}\rangle$.

1. *Do you think that this network of ontologies is well designed? Why?*

   It is correctly defined because it is made a set of ontologies and a set of alignments between these ontologies. However, the statement $\texttt{op:Buch} \sqsubseteq \texttt{opp:Roman}$ *seems strange and maybe exactly the opposite.*

2. *Is this network consistent? Provide a model for this network of ontologies.*

   The network is indeed consistent. As a model it is possible to create a model isomorphic to the ontologies (with, in each of the ontologies, the same IRI interpreted in the same way and equivalent classes having the same interpretation). A model of the network may have been a triple $\langle m, m', m''\rangle$ *such that:*

$$
\begin{aligned}
m(\texttt{o:Literature}) = m'(\texttt{op:Buch}) = m''(\texttt{opp:Roman}) &= m(\texttt{o:Novel})\\
m(\texttt{op:Poem}) &\subseteq m(\texttt{o:Literature})\\
\{m(\texttt{d:Poe}), m(\texttt{d:Mallarmé}), m'(\texttt{d:DeQuincey})\} &\subseteq m(\texttt{Writer})\\
\{m''(\texttt{d:Confessions}), m(\texttt{d:TheGoldBug})\} &\subseteq m(\texttt{o:Literature})\\
\{m(\texttt{d:TheRaven}), m(\texttt{d:Brise marine})\} &\subseteq m(\texttt{o:Poem})\\
m(\texttt{\_:b}) &= m(\texttt{d:Brise marine})\\
\langle m(\texttt{d:Poe}), m(\texttt{d:TheGoldBug})\rangle &\in m(\texttt{o:wrote})\\
\langle m(\texttt{d:Poe}), m(\texttt{d:TheRaven})\rangle &\in m(\texttt{o:wrote})\\
\langle m(\texttt{d:Mallarmé}), m(\texttt{d:Brise Marine})\rangle &\in m(\texttt{o:wrote})\\
\langle m'(\texttt{d:DeQuincey}), m'(\texttt{d:Confessions})\rangle &\in m'(\texttt{o:wrote})\\
\langle m(\texttt{d:Mallarmé}), m(\texttt{d:TheRaven})\rangle &\in m(\texttt{o:translated})\\
\langle m(\texttt{d:Baudelaire}), m(\texttt{TheGoldBug})\rangle &\in m(\texttt{o:translated})\\
\langle m''(\texttt{d:Musset}), m''(\texttt{d:Confessions})\rangle &\in m''(\texttt{o:translated})\\
\langle m'(\texttt{d:Baudelaire}), m'(\texttt{d:Confessions})\rangle &\in m'(\texttt{o:translated})
\end{aligned}
$$

3. *Provide the constraints that the alignments impose on models.*

   The constraints are that:

$$
\begin{aligned}
m(\texttt{o:Literature}) &= m'(\texttt{op:Buch})\\
m'(\texttt{op:Buch}) &\subseteq m''(\texttt{opp:Roman})\\
m''(\texttt{opp:Roman}) &= m(\texttt{o:Novel})
\end{aligned}
$$

   but since $m(\texttt{o:Novel}) \subseteq m(\texttt{o:Literature})$*, we have* $m(\texttt{o:Literature}) = m'(\texttt{op:Buch}) = m''(\texttt{opp:Roman}) = m(\texttt{o:Novel})$.

4. *What does this entail for the class (*`rdf:type`*) of* `d:Confessions` *and* `d:TheRaven` *at o in this network?*

   This entails that both works have all these four classes as `rdf:type`*. In particular,* $\langle \texttt{d:TheRaven}, \texttt{rdf:type}, \texttt{o:Poem}\rangle$ *and* $\langle \texttt{d:TheRaven}, \texttt{rdf:type}, \texttt{o:Novel}\rangle$.

Semantic peer-to-peer systems

Exercise 28 (Semantic peer-to-peer system)**.** *Given a peer-to-peer system with peers n_1, n_2 and n_3 all related to each others*[2]*. Peers n_1 and n_2 share the ontology o that has been defined at Question 1 of Exercise 8, n_3 uses the ontology o' defining;*

```
t:Work rdf:type owl:Class.
t:copyrightHolder rdf:type rdf:Property.
t:copyrightHolder rdfs:domain t:Work.
t:year rdf:type rdf:Property.
t:year rdfs:domain t:Work.
t:year rdfs:range xsd:integer.
```

There exists an alignment A between o and o' containing two correspondences: `t:Work` $\geq$ `m:Livre`, `t:year` $\equiv$ `m:annee`. Assume that n_1 does not contain instances; n_2 contains:

```
http://mm.com#a345 m:aecrit http://isbn.org/2070360423.
http://mm.com#a345 m:aecrit http://isbn.org/2070360024.
http://mm.com#a345 m:aecrit http://isbn.org/2070322882.
http://mm.com#a345 foaf:name "Albert".
http://isbn.org/2070360423 rdf:type m:Roman.
http://isbn.org/2070360423 dc:title "La peste".
http://isbn.org/2070360024 rdf:type m:Roman.
http://isbn.org/2070360024 dc:title "L'étranger".
http://isbn.org/2070322882 rdf:type m:Livre.
http://isbn.org/2070322882 dc:title "Le Mythe de Sisyphe".
...
```

and n_3 contains:

```
http://isbn.org/2070360423 t:year 1947.
http://isbn.org/2070322882 t:year 1942.
...
```

1. *Express the alignment A in OWL.*

   ```
   <owl:Class rdf:about="m:Livre">
     <rdfs:subClassOf rdf:resource="t:Work"/>
   </owl:Class>

   <owl:DataProperty rdf:about="t:year">
     <owl:equivalentProperty rdf:resource="m:annee"/>
   </owl:DataProperty>
   ```

2. *The peer n_1 would like to evaluate the following query:*

   ```
   SELECT ?t, ?y
   PREFIX ...
   WHERE
     ?x foaf:name "Albert".
     ?x m:aecrit ?l.
     ?l rdf:type m:Roman.
     ?l dc:title ?t.
     OPTIONAL { ?l m:annee ?y. }
   ```

[2] This exercise has not been given at the actual exam.

How is it possible to answer this query by using n_2, n_3 and A?

It can be evaluated directly in n_1 and n_2. It can be translated with the help of A for being evaluated in n_3. However, the three data base do not contain the same type of information: n_1 contains no data, n_2 contains information about authors and titles while n_3 contains information about publication year. Hence, instead of transforming the query for evaluating it on n_3, it is better to split it into two queries.

The result of the transformation would be:

```
SELECT ?l, ?t
PREFIX ...
WHERE
  ?x foaf:name "Albert".
  ?x m:aecrit ?l.
  ?l rdf:type m:Roman.
  ?l dc:title ?t.
```

and

```
SELECT ?l, ?y
PREFIX ...
WHERE
OPTIONAL { ?l m:annee ?y }
```

The second query can be transformed thanks to A into:

```
SELECT ?l, ?y
PREFIX ...
WHERE
OPTIONAL { ?l t:year ?y }
```

and their results can be joined. This is not particularly efficient but this should work. It is also possible to first evaluate the former query to retrieve `?l` and then to issue queries for specific values of the first answer.

3. *Provide the answer on the available data.*

   Evaluating directly the query on n_2 would yield:

   $$\{\langle \texttt{"La peste"}, \texttt{null}\rangle, \\ \langle \texttt{"L'étranger"}, \texttt{null}\rangle, \ldots\}$$

   while evaluating a transformation of this query against n_3 would yield an empty answer set.

   Using the strategy provided above would return the results:

   $$\{\langle \texttt{"La peste"}, 1947\rangle, \\ \langle \texttt{"L'étranger"}, \texttt{null}\rangle, \ldots\}$$

Bibliography

[ABITEBOUL et al. 2011] Serge ABITEBOUL, Ioana MANOLESCU, Philippe RIGAUX, Marie-Christine ROUSSET, and Pierre SENELLART, *Web data management*, Cambridge university press, Cambridge (UK), 2011.

[ALKHATEEB 2008] Faisal ALKHATEEB, Querying RDF(S) with regular expressions, en, Thèse d'informatique, Université Joseph Fourier, Grenoble (FR), 2008.

[ALKHATEEB, BAGET, et al. 2008] Faisal ALKHATEEB, Jean-François BAGET, and Jérôme EUZENAT, Constrained regular expressions in SPARQL, en, in: *Proc. international conference on semantic web and web services (SWWS), Las Vegas (NV US)*, pp. 91–99, 2008.

[ALKHATEEB, BAGET, et al. 2009] — Extending SPARQL with regular expression patterns (for querying RDF), *Journal of web semantics* 7(2):57–73, 2009.

[ALKHATEEB and EUZENAT 2013] Faisal ALKHATEEB and Jérôme EUZENAT, *Answering SPARQL queries modulo RDF Schema with paths*, Research Report 8394, INRIA, Montbonnot (FR), 2013.

[ALKHATEEB and EUZENAT 2014] — Constrained regular expressions for answering RDF-path queries modulo RDFS, *International journal of web information systems*, 2014.

[ALLEN 1983] James ALLEN, Maintaining knowledge about temporal intervals, *Communication of the ACM* 26(11):832–843, 1983.

[ANTONIOU, GROTH, et al. 2012] Grigoris ANTONIOU, Paul GROTH, Frank VAN HARMELEN, and Rinke HOEKSTRA, *A semantic web primer*, 3rd ed., The MIT press, Cambridge (MA US), 2012.

[ANTONIOU and VAN HARMELEN 2008] Grigoris ANTONIOU and Frank VAN HARMELEN, *A semantic web primer*, 2nd ed., The MIT press, Cambridge (MA US), 2008.

[ARTALE et al. 2009] Alessandro ARTALE, Diego CALVANESE, Roman KONTCHAKOV, and Michael ZAKHARYASCHEV, The DL-Lite Family and Relations, *Journal of artificial intelligence research* 36:1–69, 2009.

[ATENCIA, BORGIDA, et al. 2012] Manuel ATENCIA, Alexander BORGIDA, Jérôme EUZENAT, Chiara GHIDINI, and Luciano SERAFINI, A Formal Semantics for Weighted Ontology Mappings, in: *Proc. 11th International Semantic Web Conference (ISWC)*, vol. 7649, Lecture notes in computer science, pp. 17–33, 2012.

[ATENCIA, DAVID, et al. 2014] Manuel ATENCIA, Jérôme DAVID, and Jérôme EUZENAT, Data interlinking through robust linkkey extraction, in: *Proc. 21st European Conference on Artificial Intelligence (ECAI), Prague (CZ)*, pp. 15–20, 2014.

[BAADER et al. 2003] Franz BAADER, Diego CALVANESE, Deborah MCGUINNESS, Daniele NARDI, and Peter PATELSCHNEIDER, eds., *The description logic handbook: theory, implementations and applications*, Cambridge University Press, 2003.

[BAGET 2003] Jean-François BAGET, Homomorphismes d'hypergraphes pour la subsomption en RDF, in: *Proc. 3e journées nationales sur les modèles de raisonnement (JNMR), Paris (France)*, pp. 1–24, 2003.

[BAGET 2005] — RDF Entailment as a Graph Homomorphism, in: *Proc. 4th International Semantic Web Conference (ISWC), Galway (IE)*, pp. 82–96, 2005.

[BAGET et al. 2011] Jean-François BAGET, Michel LECLÈRE, Marie-Laure MUGNIER, and Eric SALVAT, On rules with existential variables: Walking the decidability line, *Artificial Intelligence* 175(9–10):1620–1654, 2011.

[AL-BAKRI et al. 2016] Mustafa AL-BAKRI, Manuel ATENCIA, Jérôme DAVID, Steffen LALANDE, and Marie-Christine ROUSSET, Uncertainty-Sensitive Reasoning for Inferring sameAs Facts in Linked Data, in: *Proc. 22nd European Conference on Artificial Intelligence (ECAI), The Hague (NL)*, pp. 698–706, 2016.

[BAO et al. 2009] Jie BAO, Elisa KENDALL, Deborah MCGUINNESS, and Peter PATEL-SCHNEIDER, *OWL 2 Web Ontology Language: quick reference guide*, Recommendation, `http://www.w3.org/TR/owl2-quick-reference/`, W3C, 2009.

[BARBIERI et al. 2010] Davide Francesco BARBIERI, Daniele BRAGA, Stefano CERI, Emanuele Della VALLE, and Michael GROSSNIKLAUS, C-SPARQL: a Continuous Query Language for RDF Data Streams, *International Journal on Semantic Computing* 4(1):3–25, 2010.

[BECK et al. 2015] Harald BECK, Minh DAO-TRAN, Thomas EITER, and Michael FINK, LARS: A Logic-Based Framework for Analyzing Reasoning over Streams, in: *Proc. 29th Conference on Artificial Intelligence (AAAI), Austin (TX US)*, pp. 1431–1438, 2015.

[BECKETT 2006] Dave BECKETT, *Turtle - Terse RDF Triple Language*, tech. rep., Hewlett-Packard, Bristol (UK), 2006.

[BECKETT 2009] — *OWL 2 Web Ontology Language Document Overview*, Recommendation, W3C, 2009.

[BERNERS-LEE 1998] Tim BERNERS-LEE, *Design issue: What the Semantic Web can represent*, 1998.

[BERNERS-LEE 2006] — *Design issue: Linked data*, 2006.

[BERNERS-LEE et al. 1998] Tim BERNERS-LEE, Roy FIELDING, and Larry MASINTER, *Uniform Resource Identifiers (URI): Generic Syntax*, RFC 2396, `http://www.ietf.org/rfc/rfc2396.txt`, IETF, 1998.

[BIZER et al. 2009] Christian BIZER, Jens LEHMANN, Georgi KOBILAROV, Sören AUER, Christian BECKER, Richard CYGANIAK, and Sebastian HELLMANN, DBpedia: a crystallization point for the web of data, *Journal of web semantics* 7(3):154–165, 2009.

[BOLEY et al. 2013] Harold BOLEY, Gary HALLMARK, Michael KIFER, Adrian PASCHKE, Axel POLLERES, and Dave REYNOLDS, *RIF Core Dialect*, Recommendation, `http://www.w3.org/TR/rif-core/`, W3C, 2013.

[BORGIDA and SERAFINI 2003] Alexander BORGIDA and Luciano SERAFINI, Distributed Description Logics: Assimilating Information from Peer Sources, *Journal on Data Semantics* I:153–184, 2003.

[BRICKLEY and GUHA 2004] Dan BRICKLEY and Ramanathan GUHA, *RDF Vocabulary Description Language 1.0: RDF Schema*, Recommendation, `http://www.w3.org/TR/2004/REC-rdf-schema-20040210/`, W3C, 2004.

[BROEKSTRA 2003] Jeen BROEKSTRA, SeRQL: Sesame RDF query language, in: *SWAP Delivrable 3.2 Method Design*, 2003.

[CALI et al. 2009] Andrea CALI, Georg GOTTLOB, and Thomas LUKASIEWICZ, A general Datalog-based framework for tractable query answering over ontologies, in: *Proc. 28th ACM Principle of Database Systems conference (PODS), Providence (RI US)*, pp. 77–86, 2009.

[CALVANESE, DE GIACOMO, LEMBO, et al. 2007] 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.

[CALVANESE, DE GIACOMO, and LENZERINI 2002] Diego CALVANESE, Giuseppe DE GIACOMO, and Maurizio LENZERINI, A framework for ontology integration, in: Isabel CRUZ, Stefan DECKER,

Jérôme EUZENAT, and Deborah MCGUINNESS, eds., *The emerging semantic web*, IOS Press, Amsterdam (NL), pp. 201–214, 2002.

[CALVANESE, DE GIACOMO, et al. 2004] Diego CALVANESE, Giuseppe DE GIACOMO, Maurizio LENZERINI, and Riccardo ROSATI, Logical Foundations of Peer-To-Peer Data Integration, in: *Proc. 23rd Symposium on Principles of Database Systems (PODS)*, pp. 241–251, 2004.

[CALVANESE, DE GIACOMO, LENZERINI, et al. 2000] Diego CALVANESE, Giuseppe DE GIACOMO, Maurizio LENZERINI, and Moshe VARDI, View-Based Query Processing for Regular Path Queries with Inverse, in: *Proceedings of the 19th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, pp. 58–66, 2000.

[CAPADISLI et al. 2015] Sarven CAPADISLI, Sören AUER, and Axel-Cyrille NGONGA NGOMO, Linked SDMX Data: Path to high fidelity Statistical Linked Data, *Semantic Web* 6(2):105–112, 2015.

[CAROTHERS and SEABORNE 2014] Gavin CAROTHERS and Andy SEABORNE, *RDF 1.1 N-Triples – A line-based syntax for an RDF graph*, Recommendation, `http://www.w3.org/TR/n-triples/`, W3C, 2014.

[CARROLL and KLYNE 2004] Jeremy CARROLL and Graham KLYNE, *RDF concepts and abstract syntax*, Recommendation, W3C, 2004.

[CHEIN and MUGNIER 2009] Michel CHEIN and Marie-Laure MUGNIER, *Graph-based knowledge representation*, Springer, Heildelberg (DE), 2009.

[CORBY et al. 2000] Olivier CORBY, Rose DIENG, and Cédric HÉBERT, A Conceptual Graph Model for W3C Resource Description Framework, in: *Proceedings of the International Conference on Conceptual Structures*, pp. 468–482, 2000.

[CUDRÉ-MAUROUX 2008] Philippe CUDRÉ-MAUROUX, *Emergent Semantics: Interoperability in large-scale decentralized information systems*, EPFL Press, Lausanne (CH), 2008.

[CUENCA GRAU et al. 2006] Bernardo CUENCA GRAU, Bijan PARSIA, and Evren SIRIN, Combining OWL ontologies using $\mathcal{E}$-connections, *Journal of Web Semantics* 4(1):40–59, 2006.

[DAVID et al. 2007] Jérôme DAVID, Fabrice GUILLET, and Henri BRIAND, Association Rule Ontology Matching Approach, *International journal of semantic web information systems* 3(2):27–49, 2007.

[DE BRUIJN and WELTY 2013] Jos DE BRUIJN and Chris WELTY, *RIF RDF and OWL Compatibility*, Recommendation, `http://www.w3.org/TR/rif-rdf-owl/`, W3C, 2013.

[DE LEENHEER and MENS 2008] Pieter DE LEENHEER and Tom MENS, Ontology evolution; state of the art and future directons, in: Martin HEPP, Pieter De LEENHEER, Aldo De MOOR, and York SURE, eds., *Ontology management: semantic web, semantic web services, and business applications*, Springer, New-York (NY US), chap. 5, pp. 131–176, 2008.

[DEAN and SCHREIBER 2004] Mike DEAN and Guus SCHREIBER, *OWL Web Ontology Language: reference*, Recommendation, W3C, 2004.

[DOAN et al. 2004] An-Hai DOAN, Jayant MADHAVAN, Pedro DOMINGOS, and Alon HALEVY, Ontology matching: a machine learning approach, in: Steffen STAAB and Rudi STUDER, eds., *Handbook on ontologies*, Springer Verlag, Berlin (DE), chap. 18, pp. 385–404, 2004.

[DOUSSON et al. 1993] Christophe DOUSSON, Paul GABORIT, and Malik GHALLAB, Situation Recognition: Representation and Algorithms, in: *Proc. 13th International Joint Conference on Artificial Intelligence (IJCAI), Chambéry (FR)*, pp. 166–174, 1993.

[EHRIG and EUZENAT 2005] Marc EHRIG and Jérôme EUZENAT, Relaxed Precision and Recall for Ontology Matching, en, in: *Proc. International Workshop on Integrating Ontologies at the 3rd International Conference on Knowledge Capture (K-CAP)*, pp. 25–32, 2005.

[EHRIG, STAAB, et al. 2005] Marc EHRIG, Steffen STAAB, and York SURE, Bootstrapping Ontology Alignment Methods with APFEL, in: *Proc. 4th International Semantic Web Conference (ISWC)*, vol. 3729, Lecture notes in computer science, pp. 186–200, 2005.

[EITER et al. 2008] Thomas EITER, Giovambattista IANNI, Thomas LUKASIEWICZ, Roman SCHINDLAUER, and Hans TOMPITS, Combining answer set programming with description logics for the Semantic Web, *Artificial intelligence* 172(12-13):1495–1539, 2008.

[EUZENAT (ED.) 2002] Jérôme EUZENAT (ED.), *Research challenges and perspectives of the Semantic web*, EU-NSF Strategic report, ERCIM, Sophia Antipolis (FR), 2002.

[EUZENAT 1996] Jérôme EUZENAT, HyTropes: a WWW front-end to an object knowledge management system, in: *Proc. 10th demonstration track on knowledge acquisition workshop (KAW), Banff (CA)*, (62)1–12, 1996.

[EUZENAT 2001] — Granularity in relational formalisms with application to time and space representation, *Computational intelligence* 17(4):703–737, 2001.

[EUZENAT 2002] — *Sémantique des représentations de connaissance*, Notes de cours, 2002.

[EUZENAT 2007] — Semantic precision and recall for ontology alignment evaluation, in: *Proc. 20th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 348–353, 2007.

[EUZENAT 2008] — Algebras of ontology alignment relations, en, in: *Proc. 7th International Semantic Web Conference (ISWC)*, vol. 5318, Lecture notes in computer science, pp. 387–402, 2008.

[EUZENAT 2014] — *The category of networks of ontologies*, Research report 8652, INRIA, 2014.

[EUZENAT and ROUSSET 2017] Jérôme EUZENAT and Marie-Christine ROUSSET, Semantic web, in: Pierre MARQUIS, Odile PAPINI, and Henri PRADE, eds., *A guided tour of artificial intelligence research*, Springer, Heidelberg (DE), 2017.

[EUZENAT and SHVAIKO 2013] Jérôme EUZENAT and Pavel SHVAIKO, *Ontology matching*, 2nd ed., Springer, Heidelberg (DE), 2013.

[FAGIN et al. 1995] Ronald FAGIN, Joseph HALPERN, Yoram MOSES, and Moshe VARDI, *Reasoning about knowledge*, The MIT press, Cambridge (MA US), 1995.

[FARQUHAR et al. 1995] Adam FARQUHAR, Richard FIKES, PRATT, and RICE, *Collaborative ontology construction for information integration*, tech. rep. 63, Knowledge system laboratory, Stanford university, Stanford (CA US), 1995.

[FENSEL, DECKER, et al. 1998] Dieter FENSEL, Stefan DECKER, Michael ERDMANN, and Rudi STUDER, Ontobroker: The Very High Idea, in: *Proc. 11th International Florida Artificial Intelligence Research Society Conference (FLAIRS), Sanibel Island (FL US)*, pp. 131–135, 1998.

[FENSEL, HENDLER, et al. 2003] Dieter FENSEL, James HENDLER, Henry LIEBERMAN, and Wolfgang WAHLSTER, eds., *Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential*, The MIT Press, 2003.

[FRANCONI et al. 2003] Enrico FRANCONI, Gabriel KUPER, Andrei LOPATENKO, and Luciano SERAFINI, A Robust Logical and Computational Characterisation of Peer-to-Peer Database Systems, in: *Proc. International Workshop on Databases, Information Systems and Peer-to-Peer Computing (DBISP2P) at the 29th International Conference on Very Large Data Bases (VLDB)*, pp. 64–76, 2003.

[FREKSA 1992] Christian FREKSA, Temporal reasoning based on semi-intervals, *Artificial intelligence* 54(1):199–227, 1992.

[GAL et al. 2005] Avigdor GAL, Ateret ANABY-TAVOR, Alberto TROMBETTA, and Danilo MONTESI, A Framework for Modeling and Evaluating Automatic Semantic Reconciliation, *The VLDB Journal* 14(1):50–67, 2005.

[GANDON and SCHREIBER 2014] Fabien GANDON and Guus SCHREIBER, *RDF 1.1 XML syntax*, Recommendation, `http://www.w3.org/TR/rdf-syntax-grammar/`, W3C, 2014.

[GHIDINI and GIUNCHIGLIA 2001] Chiara GHIDINI and Fausto GIUNCHIGLIA, Local Models Semantics, or Contextual Reasoning = Locality + Compatibility, *Artificial Intelligence* 127(2):221–259, 2001.

[GHIDINI and SERAFINI 1998] Chiara GHIDINI and Luciano SERAFINI, Distributed First Order Logics, in: *Proc. 2nd Conference on Frontiers of Combining Systems (FroCoS)*, pp. 121–139, 1998.

[GLIMM and OGBUJI 2013] Birte GLIMM and Chimezie OGBUJI, *SPARQL 1.1 entailment regimes*, Recommendation, http://www.w3.org/TR/sparql11-entailment, W3C, 2013.

[GOTTLOB et al. 1999] Georg GOTTLOB, Nicola LEONE, and Francesco SCARCELLO, A Comparison of Structural CSP Decomposition Methods, in: *Proc. 16th International Joint Conference on Artificial Intelligence (IJCAI'99)*, pp. 394–399, 1999.

[GRAHNE and THOMO 2003] Gösta GRAHNE and Alex THOMO, Query containment and rewriting using views for regular path queries under constraints, in: *Proc. 22nd ACM symposium on Principles of database systems (PODS)*, pp. 111–122, 2003, ISBN: 1-58113-670-6, DOI: http://doi.acm.org/10.1145/773153.773165.

[GROSOF et al. 2003] Benjamin GROSOF, Ian HORROCKS, Raphael VOLZ, and Stefan DECKER, Description logic programs: combining logic programs with description logic, in: *Proc. 12th WorldWide Web conference, Budapest (HU)*, pp. 48–57, 2003.

[GUTIERREZ et al. 2004] Claudio GUTIERREZ, Carlos HURTADO, and Alberto MENDELZON, Foundations of Semantic Web Databases, in: *Proc. 23rd ACM Symposium on Principles of Database Systems (PODS), Paris (FR)*, pp. 95–106, 2004.

[HARRIS and SEABORNE 2013] Steve HARRIS and Andy SEABORNE, *SPARQL 1.1 query language*, Recommendation, http://www.w3.org/TR/sparql11-query, W3C, 2013.

[HARTIG and PIRRÓ 2016] Olaf HARTIG and Giuseppe PIRRÓ, SPARQL with Property Paths on the Web, *Semantic Web Journal*, 2016.

[HAYES 2004] Patrick HAYES, *RDF Semantics*, Recommendation, W3C, 2004.

[HAYES and PATEL-SCHNEIDER 2014] Patrick HAYES and Peter PATEL-SCHNEIDER, *RDF Semantics*, Recommendation, http://www.w3.org/TR/rdf11-mt/, W3C, 2014.

[HEATH and BIZER 2011] Tom HEATH and Christian BIZER, *Linked Data: Evolving the Web into a Global Data Space*, Morgan & Claypool, 2011.

[HITZLER et al. 2009] Pascal HITZLER, Markus KRÖTZSCH, and Sebastian RUDOLPH, *Foundations of semantic web technologies*, Chapman & Hall/CRC, 2009.

[HORROCKS, PATEL-SCHNEIDER, BOLEY, et al. 2004] Ian HORROCKS, Peter PATEL-SCHNEIDER, Harold BOLEY, Said TABET, Benjamin GROSOF, and Mike DEAN, *SWRL: a semantic web rule language combining OWL and RuleML*, Member submission, http://www.w3.org/Submission/SWRL/, 2004.

[HORROCKS, PATEL-SCHNEIDER, and VAN HARMELEN 2003] Ian HORROCKS, Peter PATEL-SCHNEIDER, and Frank VAN HARMELEN, From SHIQ and RDF to OWL: The Making of a Web Ontology Language, *Journal of web semantics* 1(1):7–26, 2003.

[HORROCKS and SATTLER 2007] Ian HORROCKS and Ulrike SATTLER, A Tableau Decision Procedure for *SHOIQ*, *Journal of automated reasoning* 39(3):249–276, 2007.

[TER HORST 2005] Herman TER HORST, Completeness, decidability and complexity of entailment for RDF Schema and a semantic extension involving the OWL vocabulary, *Journal of Web Semantics* 3(2):79–115, 2005.

[HUSTADT et al. 2007] Ullrich HUSTADT, Boris MOTIK, and Ulrike SATTLER, Reasoning in Description Logics by a Reduction to Disjunctive Datalog, *Journal of automated reasoning* 39(3):351–384, 2007.

[HUSTADT et al. 2008] — Deciding expressive description logics in the framework of resolution, *Information and computation* 206(5):579–601, 2008.

[INANTS, ATENCIA, et al. 2016] Armen INANTS, Manuel ATENCIA, and Jérôme EUZENAT, Algebraic calculi for weighted ontology alignments, en, in: *Proc. 15th International semantic web conference (ISWC), Kobe (JP)*, pp. 360–375, 2016.

[INANTS and EUZENAT 2015] Armen INANTS and Jérôme EUZENAT, An algebra of qualitative taxonomical relations for ontology alignments, en, in: *Proc. 14th International semantic web conference (ISWC), Bethleem (PA US)*, pp. 253–268, 2015.

[JANOWICZ et al. 2015] Krzysztof JANOWICZ, Frank van HARMELEN, James HENDLER, and Pascal HITZLER, Why the Data Train Needs Semantic Rails, *AI Magazine* 36(1):5–14, 2015.

[KALFOGLOU and SCHORLEMMER 2003] Yannis KALFOGLOU and Marco SCHORLEMMER, Ontology mapping: the state of the art, *The Knowledge Engineering Review* 18(1):1–31, 2003.

[KARVOUNARAKIS et al. 2002] Gregory KARVOUNARAKIS, Sofia ALEXAKI, Vassilis CHRISTOPHIDES, Dimitris PLEXOUSAKIS, and Michel SCHOLL, RQL: A Declarative Query Language for RDF, in: *Proceedings of the 11th International Conference on the World Wide Web (WWW2002)*, 2002.

[LENZERINI 2002] Maurizio LENZERINI, Data Integration: A Theoretical Perspective, in: *Proc. 21st Symposium on Principles of Database Systems (PODS)*, pp. 233–246, 2002.

[LEVY and ROUSSET 1998] Alon LEVY and Marie-Christine ROUSSET, Combining Horn Rules and Description Logics in CARIN, *Artificial intelligence* 104(1-2):165–209, 1998.

[LIGOZAT and RENZ 2004] Gérard LIGOZAT and Jochen RENZ, What Is a Qualitative Calculus? A General Framework, in: *Proc. 8th Pacific Rim International Conference on Artificial Intelligence (PRICAI), Auckland (NZ)*, Lecture notes in computer science 3157, pp. 53–64, 2004.

[LUKE et al. 1997] Sean LUKE, Lee SPECTOR, David RAGER, and James HENDLER, Ontology-based Web Agents, in: *Proc. 1st International Conference on Autonomous Agents*, pp. 59–66, 1997.

[MANOLA and MILLER 2004] Frank MANOLA and Eric MILLER, *RDF Primer*, Recommendation, `http://www.w3.org/TR/REC-rdf-syntax/`, W3C, 2004.

[MAO et al. 2010] Ming MAO, Yefei PENG, and Michael SPRING, An adaptive ontology mapping approach with neural network based constraint satisfaction, *Journal of Web Semantics* 8(1):14–25, 2010.

[MENDELZON and WOOD 1995] Alberto MENDELZON and Peter WOOD, Finding regular simple paths in graph databases, *SIAM Journal on Computing* 24(6):1235–1258, 1995.

[MOTIK, CUENCA GRAU, et al. 2009] Boris MOTIK, Bernardo CUENCA GRAU, Ian HORROCKS, Zhe WU, Achille FOKOUE, and Carsten LUTZ, *OWL 2 Web Ontology Language: profiles*, Recommendation, `http://www.w3.org/TR/owl2-profiles/`, W3C, 2009.

[MOTIK, PATEL-SCHNEIDER, et al. 2009] Boris MOTIK, Peter PATEL-SCHNEIDER, and Bernardo CUENCA GRAU, *OWL 2 Web Ontology Language: direct semantic*, Recommendation, `http://www.w3.org/TR/owl2-direct-semantics/`, W3C, 2009.

[MOTIK and ROSATI 2010] Boris MOTIK and Riccardo ROSATI, Reconciling description logics and rules, *Journal of the ACM* 57(5), 2010.

[MUÑOZ et al. 2009] Sergio MUÑOZ, Jorge PÉREZ, and Claudio GUTIERREZ, Simple and efficient minimal RDFS, *Journal of web semantics* 7(3):220–234, 2009.

[NGOMO and AUER 2011] Axel-Cyrille Ngonga NGOMO and Sören AUER, LIMES: A Time-Efficient Approach for Large-Scale Link Discovery on the Web of Data, in: *Proc. 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 2312–2317, 2011.

[PAN et al. 2005] Rong PAN, Zhongli DING, Yang YU, and Yun PENG, A Bayesian network approach to ontology mapping, in: *Proc. 4th International Semantic Web Conference (ISWC)*, vol. 3729, Lecture notes in computer science, pp. 563–577, 2005.

[PAPAKONSTANTINOU and VASSALOS 1999] Yannis PAPAKONSTANTINOU and Vasilis VASSALOS, Query rewriting for semistructured data, in: *Proc. ACM international conference on Management of data (SIGMOD), Philadelphia (PA US)*, pp. 455–466, 1999, ISBN: 1-58113-084-8.

[PATEL-SCHNEIDER et al. 2004] Peter PATEL-SCHNEIDER, Patrick HAYES, and Ian HORROCKS, *OWL Web ontology language semantics and abstract syntax*, Recommendation, W3C, 2004.

[PÉREZ et al. 2009] Jorge PÉREZ, Marcelo ARENAS, and Claudio GUTIERREZ, Semantics and Complexity of SPARQL, *ACM transactions on database systems* 34(3):16, 2009.

[PÉREZ et al. 2010] — nSPARQL: A navigational language for RDF, *Journal of Web Semantics* 8(4):255–270, 2010.

[POGGI et al. 2008] Antonella POGGI, Domenico LEMBO, Diego CALVANESE, Giuseppe DE GIACOMO, Maurizio LENZERINI, and Riccardo ROSATI, Linking Data to Ontologies, *Journal on data semantics* 10:133–173, 2008.

[POLLERES 2007] Axel POLLERES, From SPARQL to rules (and back), in: *Proc. 16th World Wide Web Conference (WWW)*, pp. 787–796, 2007.

[PRUD'HOMMEAUX and SEABORNE 2008] Eric PRUD'HOMMEAUX and Andy SEABORNE, *SPARQL Query Language for RDF*, Recommendation, W3C, 2008.

[RAHM and BERNSTEIN 2001] Erhard RAHM and Philip BERNSTEIN, A survey of approaches to automatic schema matching, *The VLDB Journal* 10(4):334–350, 2001, ISSN: 1066-8888, DOI: `http://dx.doi.org/10.1007/s007780100057`.

[ROUSSET et al. 2006] Marie-Christine ROUSSET, Philippe ADJIMAN, Philippe CHATALIC, François GOASDOUÉ, and Laurent SIMON, SomeWhere in the Semantic Web, in: *Proc. 32nd International Conference on Current Trends in Theory and Practice of Computer Science (SofSem)*, vol. 3831, Lecture notes in computer science, pp. 84–99, 2006.

[SEABORNE 2004] Andy SEABORNE, *RDQL - A Query Language for RDF*, Member Submission, W3C, 2004.

[SHETH and LARSON 1990] Amit SHETH and James LARSON, Federated database systems for managing distributed, heterogeneous, and autonomous databases, *ACM Computing Surveys* 22(3):183–236, 1990.

[SIRIN and PARSIA 2007] Evren SIRIN and Bijan PARSIA, SPARQL-DL: SPARQL Query for OWL-DL, in: *Proc. 3rd OWL Experiences and Directions Workshop (OWLED), Innsbruck (AT)*, 2007.

[STUMME and MÄDCHE 2001] Gerd STUMME and Alexander MÄDCHE, FCA-Merge: Bottom-Up Merging of Ontologies, in: *Proc. 17th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 225–234, 2001.

[SYMEONIDOU et al. 2014] Danai SYMEONIDOU, Vincent ARMANT, Nathalie PERNELLE, and Fatiha SAÏS, SAKey: Scalable Almost Key Discovery in RDF Data, in: *Proc. 13th International Semantic Web Conference (ISWC), Riva del Garda (IT)*, pp. 33–49, 2014.

[TARSKI 1941] Alfred TARSKI, On the calculus of relations, *Journal of symbolic logic* 6(3):73–89, 1941.

[VOLZ et al. 2009] Julius VOLZ, Christian BIZER, Martin GAEDKE, and Georgi KOBILAROV, Discovering and Maintaining Links on the Web of Data, in: *Proc. 8th International Semantic Web Conference (ISWC)*, vol. 5823, Lecture notes in computer science, pp. 650–665, 2009.

[WOOLDRIDGE 2000] Michael WOOLDRIDGE, *Reasoning about rational agents*, The MIT press, Cambridge (MA US), 2000.

[XU and EMBLEY 2003] Li XU and David EMBLEY, Discovering Direct and Indirect Matches for Schema Elements. In: *Proc. 8th International Conference on Database Systems for Advanced Applications (DASFAA)*, pp. 39–46, 2003.

[YANNAKAKIS 1990] Mihalis YANNAKAKIS, Graph-theoretic methods in database theory, in: *Proc. 9th ACM Symposium on Principles of Database Systems (PODS)*, pp. 230–242, 1990.

[ZIMMERMANN 2008] Antoine ZIMMERMANN, Sémantique des réseaux de connaissances: gestion de l'hétérogénéité fondée sur le principe de médiation, Thèse d'informatique, Université Joseph Fourier, Grenoble (FR), 2008.

[ZIMMERMANN 2013] — Logical formalisms for agreement technologies, in: Sasha OSSOWSKI, ed., *Agreement technologies*, Springer Verlag, Heidelberg (DE), chap. 5, pp. 69–82, 2013.

[ZIMMERMANN and EUZENAT 2006] Antoine ZIMMERMANN and Jérôme EUZENAT, Three semantics for distributed systems and their relations with alignment composition, in: *Proc. 5th conference on International semantic web conference (ISWC), Athens (GA US)*, pp. 16–29, 2006.

[ZIMMERMANN, KRÖTZSCH, et al. 2006] Antoine ZIMMERMANN, Markus KRÖTZSCH, Jérôme EUZENAT, and Pascal HITZLER, Formalizing ontology alignment and its operations with category theory,

in: *Proc. 4th International Conference on Formal Ontology in Information Systems (FOIS)*, pp. 277–288, 2006.

[ZIMMERMANN and LE DUC 2008] Antoine ZIMMERMANN and Chan LE DUC, Reasoning on a network of aligned ontologies, in: *Proc. 2nd International conference on web reasoning and rule systems (RR), Karlsruhe (DE)*, pp. 43–57, 2008.