Jérôme Euzenat

Semantics of distributed knowledge

Lecture notes

Figure 6.2 from Luc Steels and Martin Loetzsch, 'The grounded naming game'. In Luc Steels (ed.), Experiments in cultural language evolution, 2012, Chapter 1, Reprinted with kind permission from John Benjamins Publishing Company, Amsterdam/Philadelphia. [www.benjamins.com].

This revision: 0c03e606d458828b18ad8bba59fdd4f7ccbc9e26
Compiled: September 15, 2024

1

Introduction

These are the lecture notes of a renewed course on semantics of distributed knowledge.

If, for once, I would be able to produce something not too long...

1.1 Knowledge, agents, communication, etc.

The goal of this course is to consider the semantics of distributed knowledge. That is, how to interpret and exploit knowledge found in various independent locations. Its context is the diversity and independence of data and knowledge sources in a modern interconnected world. A typical example is that of the wealth of machine-processable information provided in the semantic web and linked data. Another example may be given by several robots, physical or software, having acquired knowledge independently but expected to operate together. This may also be thought of as a metaphor of our human, and may be later hybrid, society.

The text is considered from the standpoints of *agents*. Agents are any entities able to perceive their environment, learn knowledge from it and act in their environment, in principle according to their knowledge (Figure 1.1). In the environment, there may be other agents and one specific aspect of perceiving and acting on other agents is communicating. Agents may thus be computer systems interacting through networks, robots evolving in the same space or animals and people. One important assumption considered here, called heterogeneity, is that agents do not know what is the knowledge of other agents and do not even assume that they are made the same.

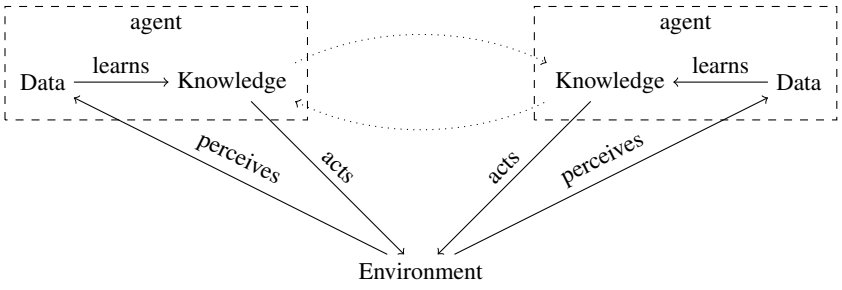


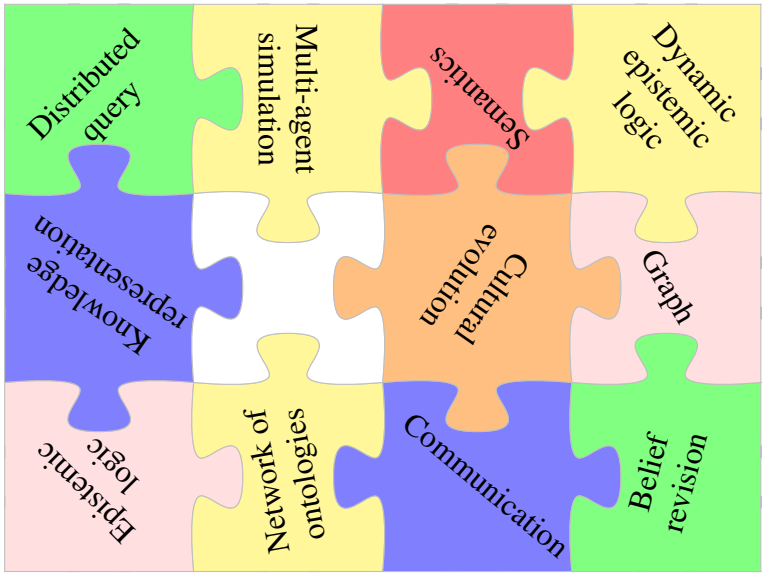
Figure 1.1. Agents and their environment.

The questions that arise in such a context are those of the interoperability of such agents. How can they operate together? Is their knowledge compatible? What knowledge do they have in common?

This course approaches such questions from several convergent perspectives which contribute answering these questions. They hold together largely due to the model-theoretic semantics designed for defining the meaning of artificial languages.

These lecture notes may be considered as a guiding line for the development of interoperable heterogeneous distributed knowledge systems. But above all, they are an invitation to apprehend rich and exciting research directions whose developments would contribute to such systems.

It will quickly sketch various approaches to distributed knowledge representation and attempt at articulating them as in the following jigsaw.



1.2 Outline/Synopsis

The course may be thought of as considering the following questions:

1. How to interpret data and knowledge?
2. How can heterogeneous knowledge be reconciliated?
3. How to query heterogeneous and distributed knowledge sources?
4. How to revise distributed knowledge when it becomes inconsistent?
5. How to socially evolve shared knowledge?
6. How to model this as action in a dynamic epistemic logic?

from a semantic standpoint. This shapes the outline of this document.

First, we have to consider how agents represent knowledge and data. In order to be understandable, the way these data are expressed has to be defined. This may consist of exposing the meaning of a column in a table or a type of relation between entities. A typical way to do this is to use a logic for defining the vocabulary used for expressing knowledge and describing data. Such a logic, through its semantics, supports reasoning engines whose inferences can be shown correct and/or complete. Chapter 2 introduces logical languages developed in the semantic web area for expressing knowledge and data.

However, different agents, designed autonomously, may use different vocabularies. This is perfectly justified as they may have had different experiences and they may deal with different problems in different environments. This heterogeneity may prevent a specific agent to interpret knowledge and data communicated by another one; think about an unknown foreign language. In order to solve this problem, it is possible to align these vocabularies by identifying which term in one language is equivalent to or more general than another one. This way, it is possible for one agent to interpret foreign knowledge. Alignments, gathering such relations, are also given a semantics allowing to increase the consequences deductible from the local knowledge. Chapter 3 describes a small language for expressing relations between logics and its semantics. This allows to create networks of ontologies.

This local knowledge may be imported from neighbours or remain distributed and gathered when necessary. The latter option is usually tackled through distributed querying. A query originating from one agent, may not

be directly answerable by itself because it lacks some data. Such a query may be answered by combining the knowledge of other agents. For that purpose, distributed query evaluation may be defined splitting the queries in subparts translated through alignments and sent to neighbour agents. The semantics of such queries, and hence the answers that it should provide, is also defined in function of that of the alignments and remaining vocabularies. Chapter 4 discusses the different strategies by which a query may be distributively evaluated and its semantics.

Within such a distributed framework, inconsistency may easily arise. In semantic terms, an inconsistency means that the expressed knowledge cannot represent any 'reality': it has no model. It may arise locally on one agent's knowledge or result from the incompatibility between the distributed knowledge of agents. In order to deal with this problem, belief revision operators are defined which attempt at restoring consistency, as soon as it occurs, through the suppression of some axioms within agents' knowledge or alignments. Chapter 5 provides the bases of belief revision and show how it can be adapted to networks of ontologies.

However, if agents' knowledge is not open to other agents, detecting inconsistencies when they occur is not always possible. On the contrary, it is possible that agents with contradictory or incorrect knowledge are able to interact in a satisfactory way for a long time before discovering such a problem. When this happens, they will take measures to adapt their knowledge with operators akin revision operators. Depending of the way they do it, it is interesting to observe how these local operators may shape the global and long term knowledge of the agent population. Chapter 6 presents experimental cultural knowledge evolution as a way for agents to evolve their knowledge through interacting with each other.

So far, we mostly considered the standpoint of a single agent connected to others and its behaviour: its knowledge, its alignments with neighbours' vocabularies, its queries, its adaptation. It is possible to semantically apprehend the evolution of the knowledge of a population of agents taking an overlooking view over the agents. This is the case of multi-agent dynamic epistemic logics able to express the knowledge and beliefs of different agents and the way it can be communicated through dynamic operators. The semantic effects of knowledge communication, such as belief revision, is embedded into the semantics of these operators. Chapter 7 introduces multi-agent dynamic epistemic logic and shows how to apply it to cultural knowledge evolution.

1.3 Lectures

When teaching this material, each chapter correspond to a three-hour session. In fact, Chapter 2 may take a bit longer, but Chapter 3 and 4 being a bit shorter, it is possible to compensate or spend more time on practical examples.

Each chapter ends with a few questions, that can be checked by the reader for understanding.

I try to provide assignments between each courses. Their purpose and structure are given in Appendix C.

1.4 Relation to previous courses

Our previous course provided a detailed account of all the technologies developed for the semantic web —data, ontology and query languages— in all their generality and variations.

Here, on the contrary, the goal is to present the bare minimal material to keep the notions meaningful, still general, yet simple. It aims at emphasising the connections between these notions, especially the connections that are maintained by the semantics. Hence, I dropped generality and completeness for simplicity. I provide minimum languages sufficient to understand problems at stake with the semantics aspect of distributing knowledge.

The notes of this previous course[1] are thus still relevant for those who want more details, but diverge largely from what will be found in these current lecture notes.

1.5 Acknowledgements

I sincerely thank the students for their patience and numerous questions.

[1] https://moex.inria.fr/teaching/sw/semwebsem.pdf

Thanks to Line van den Berg and Manuel Atencia for parts of Chapter 7 and to Yasser Bourahla and Manuel Atencia for some of the material I use when teaching Chapter 6.

Contents

2

Semantics of knowledge representation

IN WHICH WE START REPRESENTING KNOWLEDGE ON COMPUTERS, FIRST INDEPENDENTLY — ASSESSING MEANING TO THIS REPRESENTATION REQUIRES A SEMANTICS — THIS SOMETIMES REVEAL THAT OUR KNOWLEDGE IS CONTRADICTORY, REDUNDANT OR MISALIGNED.

An agent may want to record and express data observed in its environment. Arguably, this data is more useful if expressed in a regular way. This enables to identify categories of entities (e.g. Musicians, Composers, Pieces of music) entertaining precise relations (e.g. composing, performing). Moreover, such categories and relations may be further defined so that constraints hold them together (e.g. a composer has composed at least one piece of music, if someone sings a song, then it performs a piece of music). This may achieved using adequate languages.

Among humans, we use languages to communicate. A language is characterised by its syntax (lexicon and grammar) and its semantics (what a statement means). The syntax defines which are the valid statements (sometimes called well-formed formulas), the semantics if they are true or not.

For communicating with computers, we also use languages. A programming language is such a language. Programming languages also have a syntax, hence syntax errors, and a semantics which may be a mathematical denotation of programs, such as a mathematical function, or, at minima, the set of instructions the machine will perform when executing it.

But there are many others. For instance, training a machine learning system, requires a language which is made of statements. Each statement has two parts: an instance, e.g. a music file, and a label, i.e. the work performed. The two parts of the statement are the syntax of the language. The meaning of this statement is that the label is the expected output for, i.e. the work performed in, the instance.

In artificial intelligence, the field of knowledge representation defines languages for... representing knowledge. They allow a computer program, or agent, to have a representation of the knowledge that it has about the world, its goals, the other agents, etc. These representations will also be defined by providing a language and its semantics. This is the classical way of defining a logic.

I use the term ‘knowledge’ liberally, and more specifically in the tradition of ‘knowledge representation’. There are two ways of contrasting ‘knowledge’ to something else:

- In knowledge representation, ‘knowledge’ is distinguished from ‘data’. Hence, the emphasis is on opposing general statements, e.g. ‘Composers compose pieces of music’ (‘knowledge’), to particular statements, e.g. ‘Thelonius Monk has composed “Ruby, my dear”’ (‘data’).
- In philosophy, ‘knowledge’ is distinguished from ‘beliefs’, and more precisely, knowledge is considered as ‘true belief’. Hence, the emphasis is on the epistemological status of statements. For instance, one may know that ‘John Coltane has played “Ruby, my dear”’, but only believe that ‘John Coltrane has composed “Ruby, my dear”’.

Unfortunately, this course uses both acceptions.

2.1 Expressing data with RDF: syntax and semantics

The first language that we will consider is used for expressing things, or data. RDF means 'resource description framework'. It is the first language of the semantic web.

This is covered in a detailed form in [EUZENAT 2007, Chapter 2].

2.1.1 RDF Graphs

This is covered in a detailed form in [EUZENAT 2007, §2.2].

RDF can be expressed in a variety of formats. I use here its abstract syntax (triple format), which is sufficient for most purposes, and its graphical notation, RDF graphs.

RDF has been designed to describe 'resources' through formal statements. Resources are roughly anything that can be denoted by a IRI (in logic, a symbol). A resource may be a web page, may be a person, may be a concept described by a web page (e.g. 'singer'). It may be described by its class: 'singer' is a 'musician', a 'person', by its attributes, it has a 'name' or a 'vocal range'.

RDF terminology

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

This terminology is made of literals, like strings or numbers for describing entities, of names for naming entities and of placeholders to denote entities without names. "Ruby, my dear" is a string, title is an entity, this is a placeholder. 'this has for title "Ruby, my dear"' is a statement of RDF. Names correspond to logical symbols and are identified by IRIs (Internationalized Resource Identifiers, e.g. dc:title), placeholders correspond to variables (_:t1) and are called 'blanks' in RDF.

Definition 2.1 (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*, 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) or expressed as _: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). Because I want to stress the compatibility of the RDF structure with classical logic, I will use the term variable instead of that of 'blank' which is a vocabulary specific to RDF.

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

2.1.2 RDF graphs as sets of triples

RDF graphs are sets of triples $\langle s, p, o \rangle$ constructed over IRIs, blanks, and literals [CARROLL and KLYNE 2004]. In such triples, s is called the subject, p the predicate and o the object.

Definition 2.2 (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.*

The set of triples of Listing 2.1 makes the RDF graph G represented in Figure 2.1:

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

```
d2:g1          foaf:name       "Art" .
d2:g2          foaf:name       "John" .
d2:g3          foaf:name       "Thelonius" .
_:t1           mo:composer     d2:g3 .
_:t1           dc:title        "Ruby, my dear" .
d2:b1          mo:performed    _:t1 .
d2:b1          mo:member       d2:g1 .
d2:b1          mo:member       d2:g3 .
d2:b2          mo:member       d2:g2 .
d2:b2          mo:member       d2:g3 .
d2:b2          mo:performed    _:t1 .
```

Listing 2.1. RDF graph G as triples.

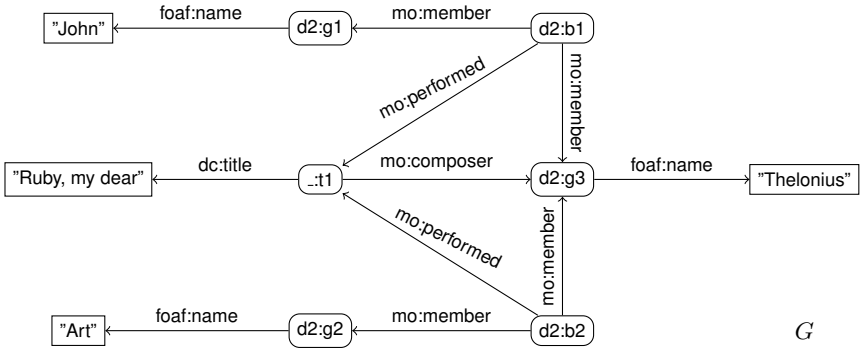


Figure 2.1. The RDF graph G of Listing 2.1.

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 the 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 2.3 (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.

The set of triples of Listing 2.2 makes the GRDF graph (H*) represented in Figure 2.2:*

```
_:g3           foaf:name       "Thelonious" .
_:b1           mo:member       _:g3 .
_:t1           _:rel           _:g3 .
_:g1           foaf:name       _:bnm .
_:b1           mo:member       _:g1 .
_:b1           mo:performed    _:t1 .
```

Listing 2.2. GRDF graph H as triples.

*Intuitively, this graph means that there exists an entity named (*foaf:name*)* "Thelonious" *who was member (*mo:member*) of a group which performed (*mo:performed*) a tune to which it is related (*_:rel*).*

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

A *GRDF graph* can be represented graphically as a directed labeled multigraph $\langle N, E, \gamma, \lambda \rangle$ such that 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 other nodes are represented by rectangles with rounded corners.

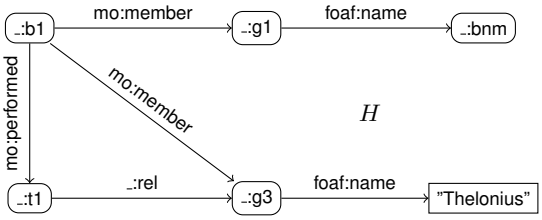


Figure 2.2. The GRDF graph H of Listing 2.2.

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

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

2.1.3 RDF Semantics

This is covered in a detailed form in [EUZENAT 2007, §2.3].

The semantics of a language may be given by defining how to interpret expressions of the language. It usually goes through providing a function which maps expressions to a particular domain. This function follows rules according to the structure of the expressions, i.e. the interpretation of 'the composer of this' depends on the interpretation of 'composer' and the interpretation of 'this'.

Of course, a computer considering an expression usually has no way to decide what it denotes. So how can it make the link with the real world? (Spoiler): it does not. The semantics does not tell *what* the interpretation of an expression is, it tells *how* to interpret an expression. Thus, it considers all the possible interpretations (called models) of a specific description of the world to decide what it entails. Of course, if the world is according to the descriptions, it will be one of these models.

Simple entailment

[HAYES 2004] introduces several semantics for RDF graphs[2]. In this section, I present only the *simple semantics* without RDF/RDFS vocabulary [BRICKLEY and GUHA 2004]. The semantics of RDF graphs is defined in model theory. RDF is thus a proper logics. The definitions of interpretations, models, satisfiability, and entailment correspond to the *simple interpretations*, *simple models*, *simple satisfiability*, and *simple entailments* of [HAYES 2004], but could be reconsidered with respect to [HAYES and PATEL-SCHNEIDER 2014].

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 domain 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 (§2.2). An interpretation with no particular extra conditions on a vocabulary (including the RDF vocabulary itself) will be simply called an interpretation.

Definition 2.4 (Interpretation of a vocabulary). *Let $V \subseteq \mathcal{V} = \mathcal{U} \cup \mathcal{L}$ be a vocabulary, an* interpretation *of V is a quadruple $I = \langle \Delta, \mathcal{P}, [\cdot], \cdot^{\iota} \rangle$ such that:*

– *Δ is a set of* resources *that contains $V \cap \mathcal{L}$;*

– *$\mathcal{P} \subseteq \Delta$ is a set of properties;*

[2] Retrospectively, a simpler semantics following strictly description logics may have been more inline with the goal to simplify everything. It is likely that my admiration about the simple trick introduced by Patrick Hayes, led me to keep it.

– $[\![\cdot]\!] : \mathcal{P} \to 2^{\Delta \times \Delta}$ *associates to each property a set of pairs of resources called the* extension *of the property;*
– *the interpretation function* $\iota : V \to \Delta$ *associates to each name in* V *a resource of* Δ*, if* $v \in \mathcal{L}$*, then* $v^\iota = v$*.*

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 ($[\![\cdot]\!]$) 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.

Models

The RDF semantics 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, s and o are interpreted as resources, and the pair of resources $\langle s^\iota, r^\iota\rangle$ belongs to the extension of the property $[\![p^\iota]\!]$. A triple $\langle s, p, ?b\rangle$ with the variable $?b \in \mathcal{B}$ would be true under I if there exists a resource $?b^\iota$ such that the pair $\langle s^\iota, ?b^\iota\rangle$ belongs to the extension $[\![p^\iota]\!]$. When interpreting a variable node, an arbitrary resource can be chosen. To ensure that a variable is always interpreted by the same resource, the interpretation function is extended as follow.

Definition 2.5 (Extension to variables). *Let* $I = \langle \Delta, \mathcal{P}, [\![\cdot]\!], \cdot^\iota\rangle$ *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* $\iota' : \mathcal{V} \cup B \to \Delta$ *such that* $\forall x \in V$*,* $x^{\iota'} = x^\iota$*.*

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

Definition 2.6 (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* $I = \langle \Delta, \mathcal{P}, [\![\cdot]\!], \cdot^\iota\rangle$ *of* V *is a* model *of* G *if there exists an extension* ι' *of* ι *to* $\mathcal{B}(G)$ *such that for each triple* $\langle s, p, o\rangle$ *of* G*,* $p^{\iota'} \in \mathcal{P}$ *and* $\langle s^{\iota'}, o^{\iota'}\rangle \in [\![p^{\iota'}]\!]$*.*

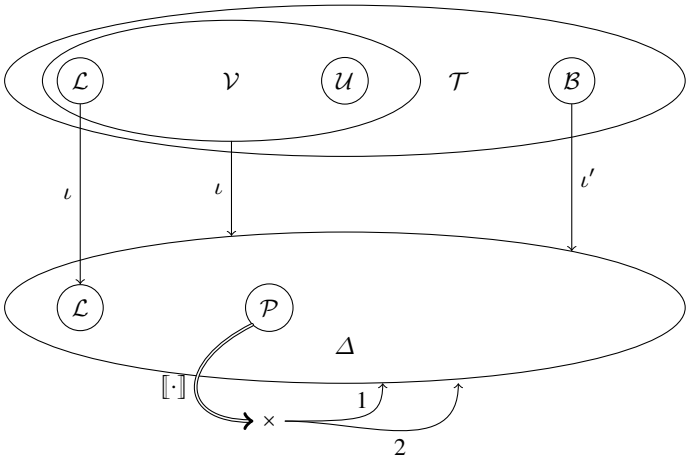


Figure 2.3. Domain structure for Simple RDF semantics.

This way of defining models unveils the specific quantification of blanks: a blank in RDF is a variable existentially quantified over a particular graph.

Satisfiability, validity, entailment and consequence

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

Definition 2.7 (Satisfiability, validity, consequence).
– *A graph G is* satisfiable *if it has a model.*
– *G is* valid *if 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 $G \models_{GRDF} G'$, if every model of G is also a model of G'.*

Proposition 2.8 (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);*

The GRDF graph H of Figure 2.2 is entailed by the RDF graph G of Figure 2.1. Moreover, the set of triples of Listing 2.3, representing an RDF graph G', also entails H:

```
_:g3    foaf:name      "Thelonious" .
_:b1    mo:member      _:g3 .
_:t1    dc:creator     _:g3 .
_:b1    mo:performed   _:t1 .
```

Listing 2.3. RDF graph G' as triples.

Indeed, in both cases, there exists a subset of triples which is an instance of H (e.g. with _:g3 *and* _:g1 *instantiated as* d2:g3, _:t1 *instantiated as* _:t1, _:b1 *instantiated as* d2:b2 *and* _:rel *instantiated as either* dc:creator *or* mo:composer*).*

However, none of G or G' entails the other.

Proposition 2.9 (Satisfiability, validity [BAGET 2005; TER HORST 2005]). *Every GRDF graph is satisfiable. The only valid GRDF graph is the empty graph.*

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 $?rel$ replaced by mo:composer is translated by:

$$\exists ?b1, ?t1, ?g1, ?g3, ?bnm : \text{foaf:name}(?g3, \text{"Thelonius"}) \wedge \text{foaf:name}(?g1, ?bnm) \wedge$$
$$\text{mo:member}(?b1, ?g1) \wedge \text{mo:member}(?b1, ?g3) \wedge \text{mo:performed}(?b1, ?t1) \wedge \text{mo:composer}(?t1, ?g3)$$

It can be checked that it is entailled by the translation of G.

The models of such a formula are isomorphic to those of the simple RDF semantics of the graph.

2.1.4 Computation

The interesting problem in Simple RDF and GRDF is SIMPLE RDF ENTAILMENT [HAYES 2004].
Problem: SIMPLE RDF ENTAILMENT
Input: two GRDF graphs G and H.
Question: Does $G \models_{\text{GRDF}} H$?

This problem is NP-complete for RDF as well as GRDF graphs. If the entailed graph is ground, then the problem becomes polynomial.

It can be computed as a kind of graph homomorphism: a projection of the node of the entailed graph to those of the entailing graph so that (a) the structure is preserved, and (b) either labels are the same or are replaced by a variable.

2.2 A simple ontology language for modelling knowledge

This is covered in a detailed form in [EUZENAT 2007, Chapter 3].

In class, I am likely to skip this section and introduce directly OWL. The opposite choice could likely have been made: retaining RDFS and skipping OWL.

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*. I will consider that an ontology is simply a set of axioms in a specific logic. I discuss the main ontology languages designed for the web: RDF Schema (§2.2) and OWL (§2.3).

What I 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, I 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. d2:Nina is of type o2:Singer.

This section focusses on RDFS as an extension of the Simple RDF language presented in Section 2.1.1. 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.

The vocabulary is first presented (§2.2.1) before the constraints RDFS puts on the interpretation of the language (§2.2.2).

2.2.1 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. o2:Singer rdfs:subClassOf o2:Musician, and relationships between properties, e.g. mo:composer rdfs:subPropertyOf dc:creator.

RDF Typing vocabulary		
rdf:type	rdf:Property	
RDFS Typing vocabulary		
rdfs:Class	rdfs:Resource	rdfs:Literal
rdfs:domain	rdfs:range	
rdfs:subClassOf	rdfs:subPropertyOf	rdfs:Datatype

Table 2.1. The RDF and RDFS Vocabularies.

RDF already introduces a few keywords in the rdf: namespace for structuring knowledge:

- ⟨d2:Nina rdf:type o2:Singer⟩ asserts that the resource d2:Nina is an instance of the class o2:Singer;
- ⟨mo:member rdf:type rdf:Property⟩ tells that mo:member 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: namespace, such as:

- ⟨o2:Singer rdf:type rdfs:Class⟩ indicating that the resource o2:Singer as for type rdfs:Class, it is thus a class.

– ⟨o2:Singer rdfs:subClassOf o2:Musician⟩ meaning that o2:Singer is a subclass of o2:Musician, thus each instance of o2:Singer, like d2:Nina, is also an instance of o2:Musician;
– ⟨mo:member rdfs:range o2:Ensemble⟩ asserts that any resource used at the extremity of an edge labelled by mo:member is an instance of the o2:Ensemble class.

Some authors have introduced ρDF [MUÑOZ, PÉREZ, and GUTIERREZ 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. I will concentrate on this fragment, because it is the semantically relevant one. The fragment of the RDFS vocabulary considered here is presented in Table 2.1. The shortcuts that may be used for each of them are given in brackets. From now on, V_{RDFS} denotes the RDFS vocabulary, i.e. all IRIs whose prefix is rdf or rdfs.

Figure 2.4 displays the RDFS graph S made of the assertions of Listing 2.4:

```
_:t1        mo:composer   d2:g3
d2:b1       mo:member     d2:g3
```

Listing 2.4. RDF Data.

with the RDF Schema of Listing 2.5:

```
foaf:Person     rdfs:subClassOf   foaf:Agent
foaf:Group      rdfs:subClassOf   foaf:Agent

dc:creator      rdfs:domain       o1:Work
dc:creator      rdfs:range        foaf:Person

foaf:member     rdfs:domain       foaf:Group
foaf:member     rdfs:range        foaf:Agent

mo:composer     rdfs:subPropertyOf  dc:creator
mo:composer     rdfs:domain       o4:MusicPiece

mo:member       rdfs:subPropertyOf  foaf:member
mo:member       rdfs:domain       o2:Ensemble
mo:member       rdfs:range        foaf:Person
```

Listing 2.5. RDF Schema as triples.

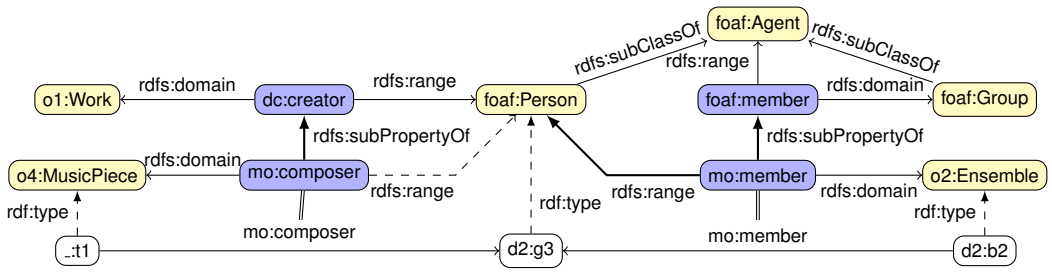


Figure 2.4. The RDFS graph S of Listing 2.5. The arrows in dashed lines are consequences of the RDFS semantics.

2.2.2 RDFS semantics

In adding new constraints to RDF interpretations, RDFS graphs may have less models, and thus more consequences. It is possible for example, in RDF, to deduce ⟨mo:Composer rdf:type rdf:Property⟩ from ⟨_:t1 mo:composer d2:g3⟩; in RDFS, to deduce ⟨_:t1 rdf:type o1:Work⟩ from ⟨_:t1 mo:composer d2:g3⟩, ⟨mo:composer rdfs:subPropertyOf dc:creator⟩, and ⟨dc:creator rdfs:domain o1:Work⟩.

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 Δ. These are interpreted in two steps, as properties (§2.1.3).

Definition 2.10 (RDFS interpretation). *An* RDFS interpretation *of a vocabulary V is a tuple $\langle \Delta, \mathcal{P}, \mathcal{C}, [\![\cdot]\!], [\cdot], \cdot^{\iota} \rangle$ such that:*

– $\langle \Delta, \mathcal{P}, [\![\cdot]\!], \cdot^{\iota} \rangle$ *is an RDF interpretation;*
– $\mathcal{C} \subseteq \Delta$ *is a distinguished subset of* Δ *identifying if a resource denotes a class of resources;*
– $[\cdot] : \mathcal{C} \to 2^{\Delta}$ *is a mapping that assigns a set of resources to every resource denoting a class;*
– $\mathcal{L} \subseteq \Delta$ *is the set of literal values,*

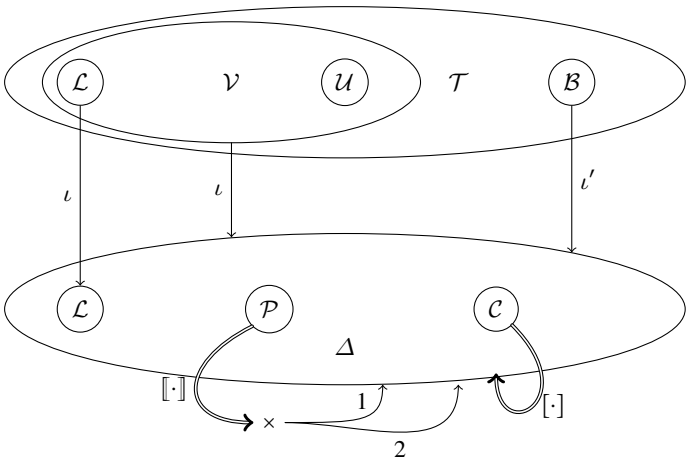


Figure 2.5. Domain structure for RDFS semantics.

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 2.11 and 2.12) as appearing in the normative semantics of RDF [HAYES 2004].

Definition 2.11 (RDF axiomatic triples). *RDF axiomatic triples are the triples in the following set:*

$$\langle \text{rdf:type}, \text{rdf:type}, \text{rdf:Property} \rangle$$

Definition 2.12 (RDFS axiomatic triples). *RDFS axiomatic triples are the triples in the following set:*

$$\langle \text{rdf:type}, \text{rdfs:domain}, \text{rdfs:Resource} \rangle \; \langle \text{rdf:type}, \text{rdfs:range}, \text{rdfs:Class} \rangle$$
$$\langle \text{rdfs:domain}, \text{rdfs:domain}, \text{rdf:Property} \rangle \; \langle \text{rdfs:domain}, \text{rdfs:range}, \text{rdfs:Class} \rangle$$
$$\langle \text{rdfs:range}, \text{rdfs:domain}, \text{rdf:Property} \rangle \; \langle \text{rdfs:range}, \text{rdfs:range}, \text{rdfs:Class} \rangle$$
$$\langle \text{rdfs:subPropertyOf}, \text{rdfs:domain}, \text{rdf:Property} \rangle \; \langle \text{rdfs:subPropertyOf}, \text{rdfs:range}, \text{rdf:Property} \rangle$$
$$\langle \text{rdfs:subClassOf}, \text{rdfs:domain}, \text{rdfs:Class} \rangle \; \langle \text{rdfs:subClassOf}, \text{rdfs:range}, \text{rdfs:Class} \rangle$$

The axiomatic triples are in principle infinite, but the simple language used here make them finite.

From this definition, any RDFS interpretation is an RDF interpretation. An RDF interpretation must be extended to identify classes. It usually corresponds to several RDFS interpretations.

Definition 2.13 (RDFS Model). *Let* G *be an RDFS graph, and* $I = \langle \Delta, \mathcal{P}, \mathcal{C}, [\![\cdot]\!], [\cdot], \cdot^{\iota} \rangle$ *be an RDFS interpretation of a vocabulary* $V \subseteq V_{RDFS} \cup \mathcal{V}$ *such that* $\mathcal{V}(G) \subseteq V$. I *is an RDFS model of* G ($I \vDash_{RDFS} G$) *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, $p^{\iota'} \in \mathcal{P}$ *and* $\langle s^{\iota'}, o^{\iota'} \rangle \in [\![p^{\iota'}]\!]$.
2. *RDF semantics:*
   a) $x \in \mathcal{P} \Leftrightarrow \langle x, \text{rdf:Property}^{\iota'} \rangle \in [\![\text{rdf:type}^{\iota'}]\!]$.
   b) I *satisfies all RDF axiomatic triples (Definition 2.11)*

3. *RDFS Classes:*
   a) if $x \in \Delta$, *then* $x \in [\![\text{rdfs:Resource}^\iota]\!]$.
   b) if $x \in \mathcal{C}$, *then* $x \in [\![\text{rdfs:Class}^\iota]\!]$.
   c) if $x \in \mathcal{L}$, *then* $x \in [\![\text{rdfs:literal}^{\iota'}]\!]$.
4. *RDFS Subproperty:*
   a) $[\![\text{rdfs:subPropertyOf}^\iota]\!]$ *is transitive and reflexive over* $\mathcal{P}$.
   b) if $\langle x, y\rangle \in [\![\text{rdfs:subPropertyOf}^\iota]\!]$ *then* $x, y \in \mathcal{P}$ *and* $[\![x]\!] \subseteq [\![y]\!]$.
5. *RDFS Subclass:*
   a) $[\![\text{rdfs:subClassOf}^\iota]\!]$ *is transitive and reflexive over* $\mathcal{C}$.
   b) if $\langle x, y\rangle \in [\![\text{rdfs:subClassOf}^\iota]\!]$, *then* $x, y \in \mathcal{C}$ *and* $[x] \subseteq [y]$.
6. *RDFS Typing:*
   a) $x \in [y]$, $\langle x, y\rangle \in [\![\text{rdf:type}^\iota]\!]$.
   b) if $\langle x, y\rangle \in [\![\text{rdfs:domain}^\iota]\!]$ *and* $\langle u, v\rangle \in [\![x]\!]$ *then* $u \in [y]$.
   c) if $\langle x, y\rangle \in [\![\text{rdfs:range}^\iota]\!]$ *and* $\langle u, v\rangle \in [\![x]\!]$ *then* $v \in [y]$.

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 2.14 (RDFS consequence). *Let* G *and* H *be two RDFS graphs, then* G RDFS-entails H *(denoted by* $G \vDash_{RDFS} H$*) 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 2.15. *If* $G \vDash_{RDF} H$ *then* $G \vDash_{RDFS} H$.

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.

Consider the graph S *described in Figure 2.4. Plain lines correspond to the triples of Listing 2.5; plain and dashed lines correspond to what can be deduced with the RDFS semantics.*

$S \nvDash_{GRDF}$ _:b rdf:type foaf:Person *because there is no subgraph of* S *that is an instance of this triple.*

But $S \vDash_{RDFS}$ _:b rdf:type foaf:Person *because for any RDFS model of* S, $\langle$_:t1$^\iota$, d2:g3$^\iota\rangle \in [\![\text{mo:composer}^\iota]\!]$ *which means that there exists an extension* ι' *of* ι *to* {_:b} *such that* $\langle$_:t1$^\iota$, _:b$^{\iota'}\rangle \in [\![\text{mo:composer}^\iota]\!]$. *By 4b, this implies that* $\langle$_:t1$^\iota$, _:b$^{\iota'}\rangle \in [\![\text{dc:creator}^\iota]\!]$ *which, by 6c, entails that* _:b$^{\iota'} \in [\![\text{foaf:Person}^\iota]\!]$. *Since this holds for every RDFS model of* S, *then* $S \vDash_{RDFS}$ _:b rdf:type foaf:Person.

The relation rdfs:subClassOf is also called 'subsumption'. It means that in all models, the interpretation of the subsumed class is included in the interpretation of the subsuming one.

2.2.3 Computation

RDFS ENTAILMENT is defined similarly as SIMPLE RDF ENTAILMENT:
Problem: RDFS ENTAILMENT
Input: two RDFS graphs G and H.
Question: Does $G \vDash_{\text{RDFS}} H$?

It is also an NP-complete problem [Gutierrez, Hurtado, and Mendelzon 2004].

In RDF, it was possible to determine if $G \vDash_{\text{RDF}} H$ by finding an RDF-homorphism between H and G (§2.1.4). In RDFS, this not so easily possible.

For determining $G \vDash_{\text{RDFS}} H$, a procedure ($\vdash$) has 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:

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

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).

2.2.4 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 in Full RDF or RDFS.

2.3 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, CALVANESE, MCGUINNESS, NARDI, and SCHNEIDER 2003], it provides constructors to constrain them precisely. The W3C has released a new version of OWL (OWL 2.0) [BECKETT 2009].

I present below the syntax, semantics and a very simplistic fragment of OWL extending that of RDFS presented in the previous section.

2.3.1 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 well-formed 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. In particular, OWL abstractly defines *class descriptions* and *property descriptions*. These can be defined recursively like terms (see Definition 2.16).

I will mostly use the XML/RDF and Turtle syntaxes [BECKETT 2009].

The OWL vocabulary is given in Table 2.2 following [DEAN and SCHREIBER 2004] for OWL and [BAO, KENDALL, MCGUINNESS, and PATEL-SCHNEIDER 2009] for OWL 2. I 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 $\langle o\ p'\ s\rangle$);

Typing vocabulary		
owl:Class	owl:Thing	owl:Nothing
rdf:Property	owl:Data(type)Property	owl:ObjectProperty
Class/Datatype$^+$ constructor		
owl:intersectionOf$^+$	owl:unionOf$^+$	owl:complementOf
	owl:oneOf$^+$	
Property restrictions		
owl:maxQualifiedCardinality^2	owl:minQualifiedCardinality^2	
owl:allValuesFrom$^+$	owl:hasValue	
Class constraints		
	rdfs:subClassOf	owl:disjointWith
Property constraints		
	rdfs:subPropertyOf	owl:inverseOf
rdfs:domain	owl:InverseFunctionalProperty	rdfs:range
Individual constraints		
rdf:type	owl:sameAs	owl:differentFrom
Property restrictions machinery		
owl:Restriction	owl:onProperty	owl:onClass^2

Table 2.2. 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: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 belongs to c (this is a universally quantified role in description logics). owl:someValuesFrom encodes existentially quantified roles.
- owl:minCardinality (resp. owl:maxCardinality) defines 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 specific class.
- owl:oneOf defines a class in comprehension by enumerating the set of its instances.
- owl:hasValue constrains a property to have (at least) a particular value (the value may in principle be a resource or a literal).
- 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.

I 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.

We provide an inductive description logic-style presentation of the OWL syntax based on the identification of IRI corresponding to individuals, properties and classes.

Definition 2.16 (OWL terms). *Given three sets of IRIs U_I, U_P and U_C, such that $\{\text{owl:Thing}, \text{owl:Nothing}, \text{rdfs:Literal}\} \subseteq U_C$, the set of object expressions is defined as:*

$$IEXP(U_I) ::= u \qquad \textit{with } u \in U_I$$

the set of property expressions is defined as:

$$PEXP(U_P) ::= u \qquad \textit{with } u \in U_P$$
$$\mid \text{owl:inverseOf}(p) \qquad \textit{with } p \in PEXP(U_P)$$

the set of class expressions is defined as:

$$
\begin{aligned}
CEXP(U_P, U_C) ::= & \; u && \text{with } u \in U_C \\
| \; & \text{owl:unionOf}(c, c') \\
| \; & \text{owl:intersectionOf}(c, c') \\
| \; & \text{owl:complementOf}(c) \\
| \; & \text{owl:oneOf}(o_1, \ldots o_n) && \text{with } o_1, \ldots o_n \in IEXP(U_I) \\
| \; & \text{owl:hasValue}(p, v) && \text{with } v \in \mathcal{L} \\
| \; & \text{owl:allValuesFrom}(p, c) \\
| \; & \text{owl:minQCardinality}(p, n, c) \\
| \; & \text{owl:maxQCardinality}(p, n, c) && \text{with } p \in PEXP(U_P),\ c, c' \in CEXP(U_P, U_C),\ n \in \mathbb{N}.
\end{aligned}
$$

OWL ontologies are based on these terms.

Definition 2.17 (OWL ontology). *Given a vocabulary V made of three sets of IRIs U_I, U_P and U_C, an ontology over V, is a set of assertions of the form:*

$$
\begin{aligned}
c \text{ rdfs:subClassOf } c' \mid c \text{ owl:disjointWith } c' \\
p \text{ rdfs:subPropertyOf } p' \mid p \text{ rdfs:domain } c \mid p \text{ rdfs:range } c \\
o \text{ rdf:type } c \mid o \text{ owl:sameAs } o' \mid o \text{ owl:differentFrom } o'
\end{aligned}
$$

with $o, o' \in IEXP(U_I)$, $p, p' \in PEXP(U_P)$, $c, c' \in CEXP(U_P, U_C)$

There exist no such grammar in OWL: OWL axioms are expressed as RDF graphs corresponding to this grammar. All assertions of Definition 2.17 are already triples so they can directly be expressed in RDF. This is not the case of all terms of Definition 2.16 which may relate more than two elements, e.g. owl:minQCardinality(p, n, c) relates p, n and c. Such expressions are rendered in RDF as resources of the class owl:Restriction. Listings 2.6 and 2.7 provide examples of this.

Below we consider an ontology O extending the RDFS graph S of Listing 2.5 with the following OWL statements:

$$
\begin{aligned}
\text{o4:MusicPiece} &\sqsubseteq \neg \text{o2:Person} \\
\text{mo:hasComposed} &\equiv \text{mo:composer}^{-1} \\
\top &\sqsubseteq \forall \text{mo:performed.o4:MusicPiece} \\
\top &\sqsubseteq \forall \text{mo:composer.o2:Person} \\
\text{mo:Pianist} &\equiv \exists_{\geq 1} \text{mo:plays.o3:Piano} \\
\text{o2:Composer} &\equiv \exists_{\geq 1} \text{mo:hasComposed.}\top \\
\text{mo:JazzTheme} &\equiv \text{o4:MusicPiece} \sqcap \exists_{\geq 1} \text{mo:performed}^{-1}\text{.o2:JazzBand}
\end{aligned}
$$

This means that:

- *No* o4:MusicPiece *is also a* o2:Person *(and vice versa);*
- mo:hasComposed *is the inverse of the* mo:composer *relation;*
- *Whatever has been* mo:performed *is a* o4:MusicPiece*;*
- *Whoever is the* mo:composer *of something is a* o2:Person*;*
- *A* mo:Pianist *is some one who* mo:plays *at least one* o3:Piano*;*
- *A* o2:Composer *is some one who* mo:hasComposed *at least one thing;*
- *A* mo:JazzTheme *is a* o4:MusicPiece *that at least one* o2:JazzBand *has* mo:performed.

Ontology O is presented in Figure 2.6.

There are several syntaxes for expressing OWL ontologies beyond the description logic syntax used above. A fragment of O can be expressed in OWL (RDF/XML) Listing 2.6:

```
<owl:Class rdf:about="o4:MusicPiece">
  <owl:disjointWith rdf:resource="o2:Person" />
```

```
</owl:Class>

<owl:ObjectProperty rdf:about="#hasComposed">
  <owl:inverseOf rdf:resource="#composer" />
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="#composer">
  <rdfs:subPropertyOf rdf:resource="dc:creator" />
  <rdfs:domain rdf:resource="o4:MusicPiece" />
</owl:ObjectProperty>

<owl:Class rdf:about="owl:Thing">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#composer" />
      <owl:allValuesFrom rdf:resource="o2:Person" />
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#performed" />
      <owl:allValuesFrom rdf:resource="o4:MusicPiece" />
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

<owl:ObjectProperty rdf:about="#performed">
  <rdfs:range rdf:resource="o4:MusicPiece" />
</owl:ObjectProperty>

<owl:Class rdf:about="#Pianist">
  <rdfs:subClassOf>
     <owl:Restriction>
      <owl:onProperty rdf:resource="#plays" />
      <owl:minQualifiedCardinality>1</owl:minQualifiedCardinality>
      <owl:onClass rdf:resource="o3:Piano" />
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:about="o2:Composer">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="hasComposed" />
      <owl:minQualifiedCardinality>1</owl:minQualifiedCardinality>
      <owl:onClass rdf:resource="owl:Thing" />
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:about="#JazzTheme">
  <rdfs:subClassOf>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Class rdf:about="o4:MusicPiece" />
        <owl:Restriction>
          <owl:onProperty>
            <owl:Property>
              <owl:inverseOf rdf:resource="mo:performed" />
            </owl:Property>
          </owl:onProperty>
          <owl:minQualifiedCardinality>1</owl:minQualifiedCardinality>
          <owl:onClass rdf:resource="o2:JazzBand" />
        </owl:Restriction>
      </owl:intersectionOf>
     </owl:Class>
  </rdfs:subClassOf>
</owl:Class>
```

Listing 2.6. RDF/XML rendering of a fragment of ontology O.

Or it can be expressed as the set of triples of Listing 2.7:

```
o4:MusicPiece    owl:disjointWith        o2:Person .

:hasComposed     owl:inverseOf           :composer .

:composer        rdfs:subPropertyOf      dc:creator .
:composer        rdfs:range              o2:Person .
```

```
:composer       rdfs:domain              o4:MusicPiece .

:performed      rdfs:domain              o4:MusicPiece .

:Pianist        rdfs:subClassOf          _:r3 .
_:r3            rdf:type                 owl:Restriction .
_:r3            owl:onProperty           :plays .
_:r3            owl:minQualifiedCardinality "1"^^xsd:Integer .
_:r3            owl:onClass              o3:Piano .

o2:Composer     rdfs:subClassOf          _:r1 .
_:r1            rdf:type                 owl:Restriction .
_:r1            owl:onProperty           :hasComposed .
_:r1            owl:minQualifiedCardinality "1"^^xsd:Integer .
_:r1            owl:onClass              o4:MusicPiece .

:JazzTheme      rdfs:subClassOf          o4:MusicPiece .
:JazzTheme      rdfs:subClassOf          _:r2 .
_:r2            rdf:type                 owl:Restriction .
_:r2            owl:onProperty           _:a1 .
_:r2            owl:minQualifiedCardinality "1"^^xsd:Integer .
_:r2            owl:onClass              o2:JazzBand .
_:a1            owl:inverseOf            :performed .
```

Listing 2.7. OWL ontology fragment O rendered as triples.

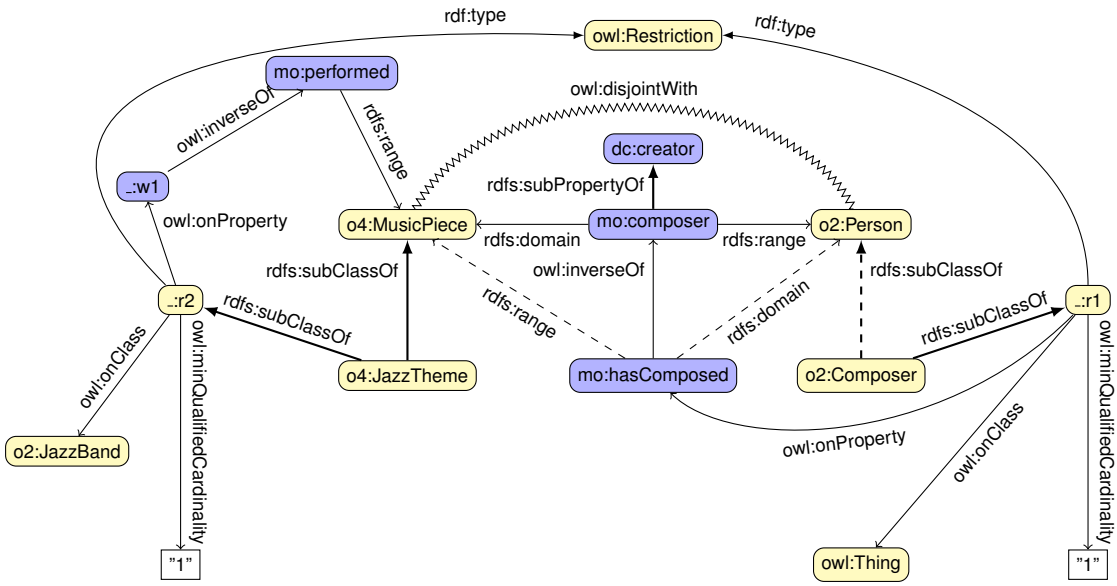


Figure 2.6. The sample OWL ontology fragment O, without the mo:Pianist definition. Dashed relations are those which can be inferred from the OWL semantics.

We extend the standard RDF graphic notation by displaying resources identified as classes in yellow and those identified as properties in blue. We also use different types of arrows for subsumption (rdfs:subClassOf, rdfs:subPropertyOf), class membership (rdf:type), and class exclusion (owl:disjointWith). This notation is summarised in Figure B.1 (p.93).

2.3.2 OWL semantics

The semantics of OWL constructs is given in [Patel-Schneider, Hayes, and Horrocks 2004; Motik, Patel-Schneider, and Cuenca Grau 2009] following a description logic style. Here I largely differs from this classical presentation (also given in [Euzenat 2007]) in order to extend the RDFS semantics.

An interpretation of an OWL vocabulary, identifies different set of entities in the domain Δ, which may be interpreted as classes or properties, more specifically:

$\mathcal{C}$ the set of classes containing owl:Thing$^\iota$ and owl:Nothing$^\iota$;

$\mathcal{D}$ the set of data types containing rdfs:Literal$^\iota$;

$\mathcal{I}$ the set of individuals;

$\mathcal{P}_d$ the set of data type properties;

$\mathcal{P}_i$ the set of object properties.

Definition 2.18 (OWL interpretation). *Let $V \subseteq \mathcal{V} = \mathcal{U} \cup \mathcal{L}$ be a vocabulary, an OWL* interpretation *of V is a tuple $I = \langle \Delta, \mathcal{C}, \mathcal{D}, \mathcal{I}, \mathcal{P}_d, \mathcal{P}_i, [\cdot], [\![\cdot]\!], \iota \rangle$ such that:*
- *$\langle \Delta, \mathcal{P}_d \cup \mathcal{P}_i, \mathcal{C} \cup \mathcal{D}, [\![\cdot]\!], [\cdot], \mathcal{L} \cap V, \cdot^\iota \rangle$ is an RDFS interpretation;*
- *Δ (resources) $\neq \emptyset$;*
- *$\mathcal{I} \subseteq \Delta$;*
- *$\mathcal{I} \neq \emptyset$;*
- *$\mathcal{L} \subseteq \Delta$;*
- *$\mathcal{I} \cap \mathcal{L} = \emptyset$;*
- *$[\cdot] : \mathcal{C} \to 2^{\mathcal{I}}$;*
- *$[\cdot] : \mathcal{D} \to 2^{\mathcal{L}}$;*
- *$[\![\cdot]\!] : \mathcal{P}_d \to 2^{\mathcal{I} \times \mathcal{L}}$;*
- *$[\![\cdot]\!] : \mathcal{P}_i \to 2^{\mathcal{I} \times \mathcal{I}}$;*
- *$\iota : \mathcal{V} \to \Delta$;*

As before an OWL interpretation can be extended to a set of variables.

Definition 2.19 (Extension of an OWL interpretation to variables). *Let $I = \langle \Delta, \mathcal{C}, \mathcal{D}, \mathcal{I}, \mathcal{P}_d, \mathcal{P}_i, [\cdot], [\![\cdot]\!], \iota \rangle$ be an OWL* interpretation *of a vocabulary $V \subseteq \mathcal{V}$, and $B \subseteq \mathcal{B}$ a set of variables. An extension of I to B is an OWL interpretation $\langle \Delta, \mathcal{C}, \mathcal{D}, \mathcal{I}, \mathcal{P}_d, \mathcal{P}_i, [\cdot], [\![\cdot]\!], \iota' \rangle$ with $\iota' : \mathcal{V} \cup B \to \Delta$ such that $\forall x \in V$, $x^{\iota'} = x^\iota$.*

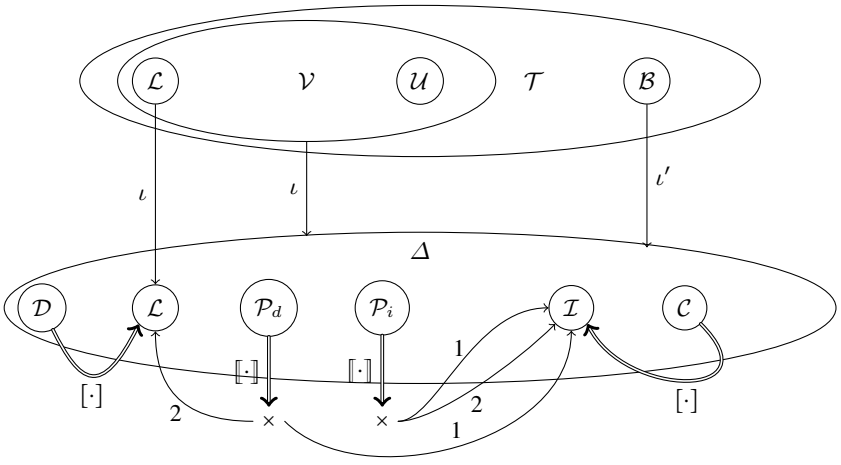


Figure 2.7. Domain structure for OWL semantics.

Compound expressions are interpreted according the classical description logic semantics.

Definition 2.20 (Interpretation of compound expressions). *Let $I = \langle \Delta, \mathcal{C}, \mathcal{D}, \mathcal{I}, \mathcal{P}_d, \mathcal{P}_i, [\cdot], [\![\cdot]\!], \iota \rangle$ be an OWL interpretation over a vocabulary $\langle \mathcal{L}, \mathcal{U} \rangle$, the interpretation of compound class expressions by I is defined by:*

$$
\begin{aligned}
[\text{owl:Nothing}^\iota] &= \varnothing \\
[\text{owl:Thing}^\iota] &= \mathcal{I} \\
[\text{rdfs:Literal}^\iota] &= \mathcal{L} \\
[u] &= [u^\iota] \\
[\![u]\!] &= [\![u^\iota]\!] \\
[\text{owl:complementOf}(c)] &= \mathcal{I} \setminus [c] \\
[\text{owl:unionOf}(c, c')] &= [c] \cup [c'] \\
[\text{owl:intersectionOf}(c, c')] &= [c] \cap [c'] \\
[\text{owl:oneOf}(o_1, \dots o_n)] &= \{o_1^\iota, \dots o_n^\iota\} \\
[\text{owl:hasValue}(p, v))] &= \{x \in \mathcal{I}; \langle x, v^\iota \rangle \in [\![p]\!]\} \\
[\text{owl:allValuesFrom}(p, c))] &= \{x \in \mathcal{I}; \forall \langle x, y \rangle \in [\![p]\!], y \in [c]\} \\
[\text{owl:minQCardinality}(p, n, c))] &= \{x \in \mathcal{I}; |\{\langle x, y \rangle \in [\![p]\!]; y \in [c]\}| \geq n\} \\
[\text{owl:maxQCardinality}(p, n, c))] &= \{x \in \mathcal{I}; |\{\langle x, y \rangle \in [\![p]\!]; y \in [c]\}| \leq n\}
\end{aligned}
$$

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

Definition 2.21 (Axiom satisfaction). *Let $I = \langle \Delta, \mathcal{C}, \mathcal{D}, \mathcal{I}, \mathcal{P}_d, \mathcal{P}_i, [\cdot], [\![\cdot]\!], \iota \rangle$ be an OWL interpretation over a vocabulary $\langle \mathcal{L}, \mathcal{U} \rangle$, an axiom δ is said to be satisfied by I (noted $I \vDash_{OWL} \delta$) as:*

$$
\begin{aligned}
I &\vDash_{OWL} \langle s, p, o \rangle \text{ if } \langle s^\iota, o^\iota \rangle \in [\![p]\!] \\
I &\vDash_{OWL} \text{rdfs:subClassOf}(c, c') \text{ if } [c] \subseteq [c'] \\
I &\vDash_{OWL} \text{owl:disjointWith}(c, c') \text{ if } [c] \cap [c'] = \varnothing \\
I &\vDash_{OWL} \text{rdfs:subPropertyOf}(p, p') \text{ if } [\![p]\!] \subseteq [\![p']\!] \\
I &\vDash_{OWL} \text{rdfs:domain}(p, c) \text{ if } \{s | \langle s, o \rangle \in [\![p]\!]\} \subseteq [c] \\
I &\vDash_{OWL} \text{rdfs:range}(p, c) \text{ if } \{o | \langle s, o \rangle \in [\![p]\!]\} \subseteq [c] \\
I &\vDash_{OWL} \text{rdf:type}(o, c) \text{ if } o^\iota \in [c] \\
I &\vDash_{OWL} \text{owl:sameAs}(o, o') \text{ if } o^\iota = o'^\iota \\
I &\vDash_{OWL} \text{owl:diferentFrom}(o, o') \text{ if } o^\iota \neq o'^\iota
\end{aligned}
$$

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 2.22 (OWL model). *An OWL interpretation $I = \langle \Delta, \mathcal{C}, \mathcal{D}, \mathcal{I}, \mathcal{P}_d, \mathcal{P}_i, [\cdot], [\![\cdot]\!], \iota \rangle$ over a vocabulary $\langle \mathcal{L}, \mathcal{U} \rangle$ is a model of an OWL Ontology O ($I \vDash_{OWL} O$) iff:*

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

The set of models of an ontology O is denoted as $\mathcal{M}(O)$.
Consequence, inconsistency, etc. are defined as usual (Appendix A).

Definition 2.23 (Consequence). *Given an ontology formula δ, δ is a consequence of an ontology O, if it is satisfied by all models of O. This is denoted as $O \vDash_{OWL} \delta$.*

Given the ontology fragment O presented in Listing 2.7, it is possible to derive various ontological consequences:

$O \vDash$ mo:performed rdfs:range o4:MusicPiece *because* $\top \sqsubseteq \forall$mo:performed.o4:MusicPiece *means that, for all models of O,* $[\![\text{mo:performed}^\iota]\!] \subseteq \Delta \times [\text{o4:MusicPiece}^\iota]$ *which means that* $O \vDash$ mo:performed rdfs:range o4:MusicPiece.

$O \models$ mo:hasComposed rdfs:domain o2:Person *because* $O \models$ mo:composer rdfs:range o2:Person *using the same reasoning as above. Moreover,* mo:hasComposed *is the converse of (*owl:inverseOf*)* mo:composer*, then it has the inverse domain and range. Indeed,* $[\![\text{mo:composer}^\iota]\!] \subseteq [\text{o4:MusicPiece}^\iota] \times [\text{o2:Person}^\iota]$ *is equivalent to* $[\![\text{mo:hasComposed}^\iota]\!] = [\![(\text{mo:composer}^{-1})^\iota]\!] = [\![\text{mo:composer}^\iota]\!]^{-1} \subseteq [\text{o2:Person}^\iota] \times [\text{o4:MusicPiece}^\iota]$.

$O \models$ o2:Composer rdfs:subClassOf o2:Person *because all models of* O *satisfy* $[\text{o2:Composer}^\iota] \subseteq [\exists_{\geq 1}\text{mo:hasComposed}.\top]$*, which means that,* $\forall o \in [\text{o2:Composer}^\iota]$ *there exists* $o' \in [\top^\iota] = \Delta$ *such that* $\langle o, o' \rangle \in [\![\text{mo:hasComposed}^\iota]\!]$*. But* $[\![\text{mo:hasComposed}^\iota]\!] \subseteq [\text{o2:Person}^\iota] \times [\text{o4:MusicPiece}^\iota]$*. This means that* $[\text{o2:Composer}^\iota] \subseteq [\text{o2:Person}^\iota]$*, hence* $O \models$ o2:Composer $\sqsubseteq$ o2:Person.

$O \models$ o2:Composer owl:disjointFrom o4:JazzTheme *because for all models of* O*,* $[\text{o4:JazzTheme}^\iota] \subseteq [\text{o4:MusicPiece}^\iota]$ *and* $[\text{o2:Composer}^\iota] \subseteq [\text{o2:Person}^\iota]$ *and* $[\text{o2:Person}^\iota] \cap [\text{o4:MusicPiece}^\iota] = \varnothing$*, then* $[\text{o4:jazzTheme}^\iota] \cap [\text{o2:Composer}^\iota] = \varnothing$*, hence* $O \models$ o2:Composer owl:disjointFrom o4:JazzTheme.

$O \models \exists_{\geq 1}\text{mo:performed}^{-1}.\text{o2:JazzBand} \sqsubseteq \text{o4:MusicPiece}$ *(showing that the definition of* mo:JazzTheme *is redundant) because for all models of* O*,* $[\exists_{\geq 1}\text{mo:performed}^{-1}.\text{o2:JazzBand}] \subseteq \{o \in \Delta | \#\{\langle o, y \rangle \in [\![(\text{mo:performed}^{-1})^\iota]\!] | y \in [\text{o2:JazzBand}^\iota]\} \geq 1\} = \{o \in \Delta | \#\{\langle y, o \rangle \in [\![(\text{mo:performed})^\iota]\!] | y \in [\text{o2:JazzBand}^\iota]\} \geq 1\}$*. This means that these objects are in the range of at least one* mo:performed *relation, but* $[\![\text{mo:performed}^\iota]\!] \subseteq \Delta \times [\text{o4:MusicPiece}^\iota]$*, hence* $[\exists_{\geq 1}\text{mo:performed}^{-1}.\text{o2:JazzBand}] \subseteq [\text{o4:MusicPiece}^\iota]$.

Some of these statements are displayed in dashed in Figure 2.6.

If one now considers the graph G'' as the graph of Figure 2.1 to which are added the following statements:

```
d2:g3          mo:plays       _:p1 .
_:p1           rdf:type       o3:Piano .
d2:b2          rdf:type       o2:JazzBand .
```

Listing 2.8. Triples added to RDF graph G to make graph G''.

with the ontology O of Listing 2.7, it is possible to deduce that:

$O \cup G \models$ d2:g3 mo:hasComposed _:t1 *because in all models of* $O \cup G$*,* $\langle \text{\_:t1}^{\iota'}, \text{d2:g3}^\iota \rangle \in [\![(\text{mo:composer}^\iota)]\!]$ *and* $[\![(\text{mo:hasComposed}^\iota)]\!] = [\![(\text{mo:composer}^{-1})^\iota)]\!] = [\![\text{mo:composer}^\iota]\!]^{-1}$*, thus* $\langle \text{d2:g3}^\iota, \text{\_:t1}^{\iota'} \rangle \in [\![(\text{mo:hasComposed}^\iota)]\!]$.

$O \cup G \models$ d2:g3 rdf:type mo:Composer *because in all models of* $O \cup G$*,* $\langle \text{d2:g3}^\iota, \text{\_:t1}^{\iota'} \rangle \in [\![\text{mo:hasComposed}^\iota]\!]$ *and* $\text{\_:p1}^{\iota'} \in [\top] = \Delta$*, thus* $\text{d2:g3}^\iota \in \{o \in \Delta | \#\{\langle o, y \rangle \in [\![\text{mo:plays}^\iota \wedge y \in \Delta]\!]\} \geq 1\}$*, hence* $\text{d2:g3}^\iota \in [\text{mo:Composer}^\iota]$.

$O \cup G'' \models$ d2:g3 rdf:type mo:Pianist *because in all models of* $O \cup G''$*,* $\langle \text{d2:g3}^\iota, \text{\_:p1}^{\iota'} \rangle \in [\![\text{mo:plays}^\iota]\!]$ *and* $\text{\_:p1}^{\iota'} \in [\text{o3:Piano}^\iota]$*, thus* $\text{d2:g3}^\iota \in \{o \in \Delta | \#\{\langle o, y \rangle \in [\![\text{mo:plays}^\iota]\!] | y \in [\text{o3:Piano}^\iota]\} \geq 1\}$*, hence* $\text{d2:g3}^\iota \in [\text{mo:Pianist}^\iota]$.

$O \cup G'' \models$ _:t1 rdf:type mo:JazzTheme *because in all models of* $O \cup G''$*,* $\langle \text{d2:b2}^\iota, \text{\_:t1}^{\iota'} \rangle \in [\![\text{mo:performed}^\iota]\!]$ *and* $\text{d2:b2}^\iota \in [\text{o2:JazzBand}^\iota]$*, thus* $\text{\_:t1}^{\iota'} \in \{o \in \Delta | \#\{\langle y, o \rangle \in [\![\text{mo:performed}^\iota]\!] | y \in [\text{o2:JazzBand}^\iota]\} \geq 1\}$*, and then* $\text{\_:t1}^{\iota'} \in \{o \in \Delta | \#\{\langle o, y \rangle \in [\![\text{mo:performed}^\iota]\!]^{-1} | y \in [\text{o2:JazzBand}^\iota]\} \geq 1\}$*, hence* $\text{\_:t1}^{\iota'} \in [\text{mo:JazzTheme}^\iota]$.

The consequence function $Cn^\omega(O) = \{\delta \mid O \models_{\text{OWL}} \delta\}$ identifies the set of consequences of Ontology O. It satisfies the three properties of a closure function (Appendix A).

2.3.3 Computation

OWL ENTAILMENT is defined as usual:
Problem: OWL ENTAILMENT
Input: an OWL ontology O, an RDFS graphs G and an OWL statement δ.
Question: Does $O, G \models_{\text{OWL}} \delta$?

But it is indeed more complicated than as usual.

The complexity of the problem may range depending on the chose version of OWL from polynomial to undecidable (OWL Full).

Conclusion

In this chapter, we have introduced various concepts to express knowledge and their semantics:

1. A graph language for expressing data: $G \ni \langle s, p, o \rangle$ (RDF triple)
2. With a semantics for graphs: $G \vDash_{GRFD} \langle s, p, o \rangle$, $G \vDash_{GRFD} G'$
3. A language for defining ontologies: $O \ni \delta$ (axiom)
4. A semantics for ontologies: $O \vDash_{\text{OWL}} \delta$, $O \vDash_{\text{OWL}} O'$, $O, G \vDash_{\text{OWL}} \langle s, p, o \rangle$, $O, G \vDash_{\text{OWL}} G'$,

Quiz

- Can the use of an (OWL) ontology change the interpretation of (RDF) data?
- Does adding more triples (axioms, correspondences) to a graph (ontology, alignment) increases its set of models?
- Does an RDF graph necessarily entails its subgraphs?
- Does an RDF graph entails its instances, i.e. graphs in which some variables are replaced by values (or other variables)?
- Does adding an ontology to a graph entails more consequences?
- How are the blank nodes, a.k.a. variables, interpreted in RDF?

References

[ANTONIOU and VAN HARMELEN 2008] Grigoris Antoniou, Frank van Harmelen, A semantic web primer, The MIT press, 2004 (rev. 2008; 2012)

[HITZLER, KRÖTZSCH, and RUDOLPH 2009] Pascal Hitzler, Markus Krötzsch, Sebastian Rudolph, Foundations of semantic web technologies, Chapman & Hall/CRC, 2009

There are many technologies implementing the principles presented here. Today, the mainstream languages are those recommended by the W3C: RDF [SCHREIBER and RAYMOND 2014] and OWL [BECKETT 2009].

3

Networks of aligned ontologies

IN WHICH WE LAY BRIDGES (ALIGNMENTS) BETWEEN THEORIES TO ADDRESS MISUNDERSTANDING, ENDING WITH A NETWORK OF RELATED ONTOLOGIES — WE PROVIDE DIFFERENT SEMANTICS FOR SUCH NETWORKS ALLOWING TO UNDERSTAND THE BENEFIT OF ALIGNMENTS FOR INTEROPERATION — WE CAN INTERPRET THE NETWORK AS A WHOLE OR IN THE NEIGHBORHOOD OF A PARTICULAR AGENT.

Agents do not evolve in a vacuum. They interact with each other so that they can use each other's knowledge. They may also take advantage of knowledge offered by other knowledge and data sources, such as data exposed on the web, e.g. wikidata, data streams generated by simple sensors, or provided by human beings or other agents.

So far, knowledge was expressed in a form which is suitable for computers and a way to assign it meaning has been introduced. However, there is no a priori reason that knowledge and data is expressed the same way.

Different ontologies may represent, differently, the same set of objets, considering them from different standpoints, e.g. musicological, historical, economical. They may also be complementary because they focus on a subset of the objects, e.g. traditional instruments, opera, punk bands. The knowledge of the traditional instrument factor has no reason to be the same as that of the classical music audiophile which is different from that of a heavy metal record seller.

They may also be complementary when they represent different sets of objects (people, work of art, devices), that may entertain some relations, like musician performing music pieces and playing music instruments.

But this heterogeneity should not prevent from interpreting 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. For a specific task, it may be useful to jointly use several of these ontologies together.

Heterogeneity may come from knowledge and data expressed:
- in different knowledge representation language (syntactic heterogeneity),
- with different vocabularies (terminological heterogeneity), or
- with different assertions (semantic heterogeneity).

Here, data is restricted to RDF and knowledge to OWL. Hence syntactic heterogeneity is not discussed.

The problem to solve is how to make sense of knowledge and data from heterogeneous sources. How to interpret them.

This problem can be solved by imposing to the world a unique knowledge representation language, a unique vocabulary and a unique ontology. However, this is both unrealistic and non desirable.

One could also consider putting all knowledge and data together. With respect to what has been seen in the previous chapter, this corresponds to taking the set of graphs $\{G_i\}_{i\in I}$ and their ontologies $\{O_i\}_{i\in I}$ and considering them as one graph $\bigcup_{i\in I} G_i$ and one ontology $\bigcup_{i\in I} O_i$. This should lead to reduce the set of models of the graphs and thus to be more informative. In theory this is possible, however this raises several problems:
- if the graphs use disjoint vocabularies, the set of models is not usefully reduced;
- the union of the graphs may be inconsistent (the set of models is empty);
- not everyone may want to provide their knowledge and not everyone has the capability to perform this.

One less drastic way to deal with this problem consists of connecting ontologies 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 chapter is to provide a formal account of such networks of ontologies. In particular, it establishes the semantics of alignments and the constraints this raises on the interpretation of distributed ontologies.

I first precisely define what are alignments through their syntax (§3.1.1) and semantics (§3.1.2) before introducing networks of ontologies and their semantics (§3.2).

3.1 Connecting ontologies with alignments

Relations between data and knowledge may be mostly based on relations between their ontologies. The process of finding the relations between ontologies is called *ontology matching* and the result of this process expressing declaratively these relations is called an *(ontology) alignment* [EUZENAT and SHVAIKO 2013]. In addition, finding the manifestation of the same object withing different data sets is called *data interlinking*.

3.1.1 Ontology alignments

This is covered in a detailed form in [EUZENAT 2007, Chapter 8] and [EUZENAT and SHVAIKO 2013].

Alignments express the correspondences between entities belonging to different ontologies. All definitions here are given for alignments between two ontologies. The definitions can be straightforwardly extended by using n-ary correspondences. A correspondence must consider the two corresponding entities and the relation that is supposed to hold between them.

Since the related entities are an important part of alignments, they have to be defined. The matched entities are connected but equivalent to those of the ontology language because it can be desirable to have a different language for identifying the matched entities. Given an ontology language, the term *entity language* denotes 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.

Definition 3.1 (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.*

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.

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* ($\geq$). They can be used without reference to any ontology language.

With these ingredients, it is possible to define the correspondences that have to be found by matching algorithms.

Definition 3.2 (Correspondence). *Given two ontologies O and O' with associated entity languages $Q_L(O)$ and $Q_{L'}(O')$ and a set of alignment relations Θ, a* correspondence *is a triple of $Q_L(O) \times \Theta \times Q_{L'}(O')$:*

$$\langle e, r, e' \rangle,$$

also noted

$$e \; r \; e'$$

The correspondence $\langle e, r, e' \rangle$ asserts that the relation r holds between the ontology entities e and e'. In the following, for the sake of simplicity, I will restrict the presentation to alignments between OWL ontologies with the entity language restricted to IRIs of the mentioned classes properties and possibly individuals, and the correspondence relations $\theta = \{\leq, \geq, \bot, =, \in, \ni\}$. = is aimed at being used between individual instances: it is quite commonly used in the context of linked data for expressing equivalence links (owl:sameAs) rather than ontology alignments. It may also be used as a shortcut for two correspondences with relations $\leq$ and $\geq$. $\in$ and $\ni$ relate individuals to classes of other ontologies. This is very uncommonly used, but it is useful within later chapters. However, results will also apply to other relations.

For example, a simple kind of correspondence is as follows:

o2:Musician $\leq$ foaf:Person

It asserts that what is denoted by the https://example.org/o2#Musician *IRI is more specific than what is denoted by* http://xmlns.com/foaf/0.1/Person*. Another example:*

dc:creator = mo:author

asserts that what is denoted by the http://purl.org/dc/elements/1.1/creator *IRI is equivalent to what is denoted by* http://purl.org/ontology/mo/author*. Hence, this relation may apply between different types of entities (here classes or properties) with different relations (= or $\leq$).*

Finally, an alignment is defined as a set of correspondences between the same pair of ontologies.

Definition 3.3 (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 3.1 displays a possible alignment for a pair of ontologies. On the left-hand side is an ontology rather focussing on selling (music) files and on the right-hand side a classification of music compositions following the (US) library of congress classification. We make the simplification that these two classifications are about the same objects (music pieces).

It can be expressed by the set of correspondences (alignment) of Listing 3.1:

cd:MP3 $\leq$ lc:Music	rdf:label $\leq$ dc:title	rdf:label $\geq$ dc:title
cd:Ragtime $\leq$ lc:Piano	cd:BigBand $\leq$ lc:Orchestral	cd:Symphony $\leq$ lc:Orchestra
cd:Forró $\leq$ lc:Trios	cd:Forró $\leq$ lc:Secular	cd:Blues $\leq$ lc:Secular
cd:Gospel $\leq$ lc:Sacred		

Listing 3.1. Alignment of Figure 3.1.

It shows the inclusion of fragmentary genre of music used in the left-hand ontology into the forms of the right-hand side ontology. Some types of music, e.g. cd:Forró*, may be found in different types of music description (*lc:Trios *and* lc:Secular *which covers all traditional songs). Conversely, some forms are used in various genres: here* lc:Orchestra *can be found in* cd:Classical *and* cd:Jazz*.*

Such correspondences may be useful for someone wanting to buy interpretation of compositions that can be found in the catalogue of the library of congress.

*In principle, the composers or lyricists of music pieces are not necessarily the same as their performers although they have related types (*o2:Musician *and* foaf:Person*).*

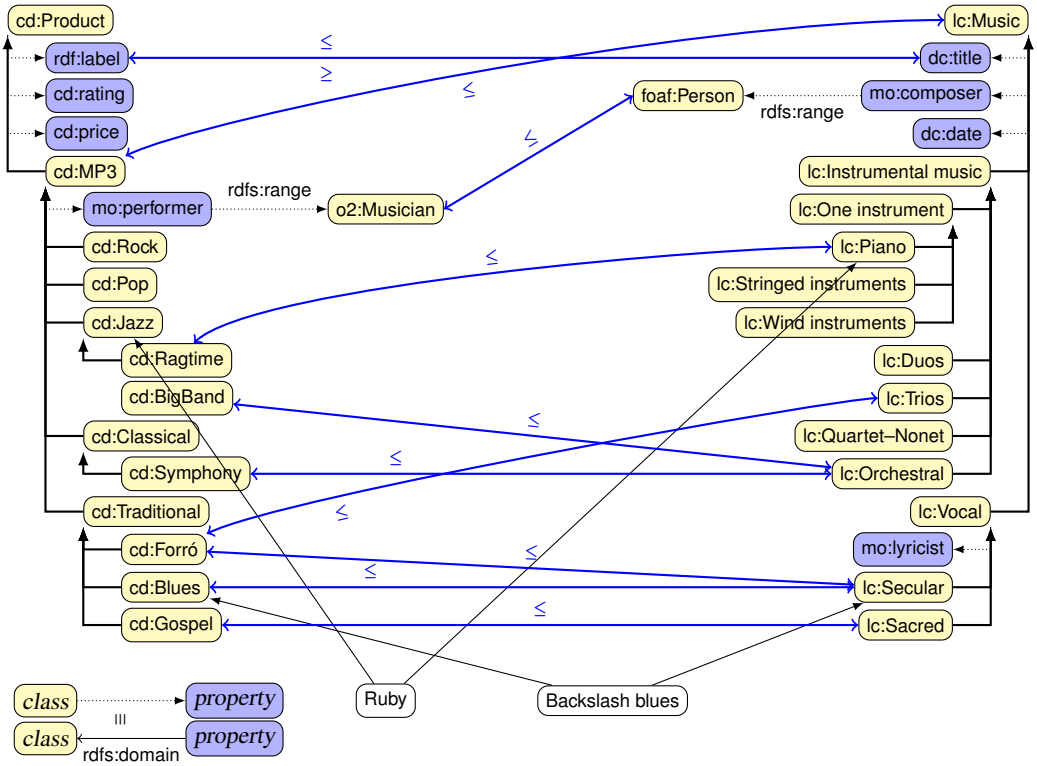


Figure 3.1. Alignment between two fragments of ontologies. Correspondences are expressed by double blue arrows.

3.1.2 Semantics of ontology alignments

This is covered in a detailed form in [EUZENAT 2007, Chapter 10 and 11].

If alignments have to be used with ontologies, they have to be given a semantics. Intuitively, an alignment expresses constraints between ontologies. It thus restricts the semantics of these ontologies to satisfy these constraints. The general semantic framework considered here will restrict the ontology models to those compatible with the alignments.

The semantics of aligned ontologies must remain compatible with the classical semantics of ontologies: connecting ontologies to other ontologies should not radically change the manner to interpret them. Indeed, 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 here is a map m from the entities of an ontology to a particular domain of interpretation. Such models have to apply to all the elements of the entity languages used in the alignments.

There is no 'standard' semantics for ontology alignments. 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, PARSIA, and SIRIN 2006] domains. These models have been compared in [ZIMMERMANN and EUZENAT 2006]. I provide here a general semantics for ontology alignment which reconciliates different semantics that have been provided by different authors. Then I provide an example with the most simple one, the reduced semantics.

For that purpose, each correspondence is interpreted with respect to models of each ontology [ZIMMERMANN 2013]. This interpretation 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.

Different semantics may be provided for alignments by defining differently $\models_\Delta$ which specifies the satisfaction of a correspondence μ by two models of the ontologies O and O' (which is denoted by $m_O, m_{O'} \models_\Delta \mu$)

However, this course, only uses the reduced semantics. The reduced alignment semantics considers that the entities of the different ontologies must be interpretable in the same domain of interpretation Δ. The semantics thus simply interpret the correspondence relations over these (sets of) entities.

Definition 3.4 (Reduced interpretation of correspondences). *In the language used as example, c and c' stand for class names, o and o' for individual names, and p and p' for relation names. If m_O and $m_{O'}$ are respective models of O and O':*

$$
\begin{aligned}
&m_O, m_{O'} \models_\Delta \langle c, \leq, c' \rangle \text{ iff } [c^{\iota_O}] \subseteq [c'^{\iota_{O'}}] \\
&m_O, m_{O'} \models_\Delta \langle c, \geq, c' \rangle \text{ iff } [c^{\iota_O}] \supseteq [c'^{\iota_{O'}}] \\
&m_O, m_{O'} \models_\Delta \langle c, \perp, c' \rangle \text{ iff } [c^{\iota_O}] \cap [c'^{\iota_{O'}}] = \varnothing \\
&m_O, m_{O'} \models_\Delta \langle p, \leq, p' \rangle \text{ iff } [\![p^{\iota_O}]\!] \subseteq [\![p'^{\iota_{O'}}]\!] \\
&m_O, m_{O'} \models_\Delta \langle p, \geq, p' \rangle \text{ iff } [\![p^{\iota_O}]\!] \supseteq [\![p'^{\iota_{O'}}]\!] \\
&m_O, m_{O'} \models_\Delta \langle o, \in, c' \rangle \text{ iff } o^{\iota_O} \in [c'^{\iota_{O'}}] \\
&m_O, m_{O'} \models_\Delta \langle c, \ni, o' \rangle \text{ iff } o'^{\iota_{O'}} \in [c^{\iota_O}] \\
&m_O, m_{O'} \models_\Delta \langle o, =, o' \rangle \text{ iff } o^{\iota_O} = o'^{\iota_{O'}}
\end{aligned}
$$

Hence, the semantics of two aligned ontologies may be given as a set of models which are pairs of compatible models.

Definition 3.5 (Models of alignments). *Given two ontologies O and O' and an alignment A between these ontologies, a model of this alignment is a pair $\langle m_O, m_{O'} \rangle \in \mathcal{M}(O) \times \mathcal{M}(O')$, such that $\forall \mu \in A$, $m_O, m_{O'} \models_\Delta \mu$.*

This is noted $m_O, m_{O'} \models_\Delta A$.

Correspondences in the alignment impose constraints to the set of acceptable models of the related ontologies, i.e. they select those models which are compatible with the models of the other ontology through the alignment. Since they reduce the set of models of an aligned ontology, they increase the set of consequences. Hence, connecting two ontologies together brings more consequences: it provides more information.

It is possible to define the consequence relation between an alignment and a correspondence as $A \models_\Delta \mu$ iff $\forall \langle m_O, m_{O'} \rangle$ such that $m_O, m_{O'} \models_\Delta A$, $m_O, m_{O'} \models_\Delta \mu$. This notation can be extended to $A \models_\Delta A'$ iff $\forall \mu \in A'$, $A \models \mu$.

Similarly as for ontologies, the semantics of alignments can be given by the relation $\models_\Delta$. The corresponding closure function is $Cn^\alpha(A) = \{\mu \mid A \models_\Delta \mu\}$.

3.2 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, KUPER, LOPATENKO, and SERAFINI 2003]), i.e. sets of ontologies and alignments.

Definition 3.6 (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.

The set of alignments in Λ between O and O' is denoted by $\Lambda(O, O')$.

Figure 3.2 presents four simplified 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 $i \sqsubset c$ denotes membership of individual i to concept c):

$$O_1 = \left\{ \begin{array}{c} \text{foaf:Agent} \sqsubseteq \top_1, \text{o1:Work} \sqsubseteq \top_1, \text{o1:Device} \sqsubseteq \top_1, \\ \text{foaf:Person} \sqsubseteq \text{foaf:Agent}, \text{foaf:Group} \sqsubseteq \text{foaf:Agent}, \text{o1:Band} \sqsubseteq \text{foaf:Group}, \\ \text{o1:Person} \bot \text{o1:Band}, \text{o1:Person} \bot \text{o1:Device}, \text{o1:Person} \bot \text{o1:Band}, \\ \text{o1:Band} \bot \text{o1:Device}, \text{o1:Band} \bot \text{o1:Work}, \text{o1:Device} \bot \text{o1:Work} \end{array} \right\}$$

$$O_2 = \left\{ \begin{array}{c} \text{o2:Ensemble} \sqsubseteq \top_2, \text{o2:Person} \sqsubseteq \top_2, \text{o2:Composer} \sqsubseteq \text{o2:Person}, \\ \text{o2:Musician} \sqsubseteq \text{o2:Person}, \text{o2:Pianist} \sqsubseteq \text{o2:Musician}, \text{o2:Singer} \sqsubseteq \text{o2:Musician} \end{array} \right\}$$

$$O_3 = \left\{ \begin{array}{c} \text{o3:MusicPerformingDev} \sqsubseteq \text{o3:Device}, \text{o3:MusicrecordingDevice} \sqsubseteq \text{o3:Device}, \\ \text{o3:MusicInstrument} \sqsubseteq \text{o3:MusicPerformingDev}, \text{o3:MusicPlayer} \sqsubseteq \text{o3:MusicPerformingDev}, \\ \text{o3:Phone} \sqsubseteq \text{o3:MusicPerformingDev}, \text{o3:Phone} \sqsubseteq \text{o3:MusicRecordingDev}, \\ \text{o3:Voice} \sqsubseteq \text{o3:MusicInstrument}, \text{o3:Haegum} \sqsubseteq \text{o3:MusicInstrument} \end{array} \right\}$$

$$O_4 = \left\{ \text{o4:Song} \sqsubseteq \text{o4:MusicPiece}, \text{o4:Theme} \sqsubseteq \text{o4:MusicPiece} \right\}$$

together with four alignments $A_{1,2}$, $A_{2,3}$, $A_{1,3}$, *and* $A_{1,4}$ *which can be described as follows:*

$$A_{1,2} = \left\{ \text{o1:Band} \geq \text{o2:Ensemble}, \text{o1:Person} \geq \text{o2:Person} \right\}$$
$$A_{2,3} = \left\{ \text{o2:Singer} \leq \text{o3:Voice} \right\}$$
$$A_{1,3} = \left\{ \text{o1:Device} \geq \text{o3:Device} \right\}$$
$$A_{1,4} = \left\{ \text{o1:Work} \geq \text{o4:MusicPiece} \right\}$$

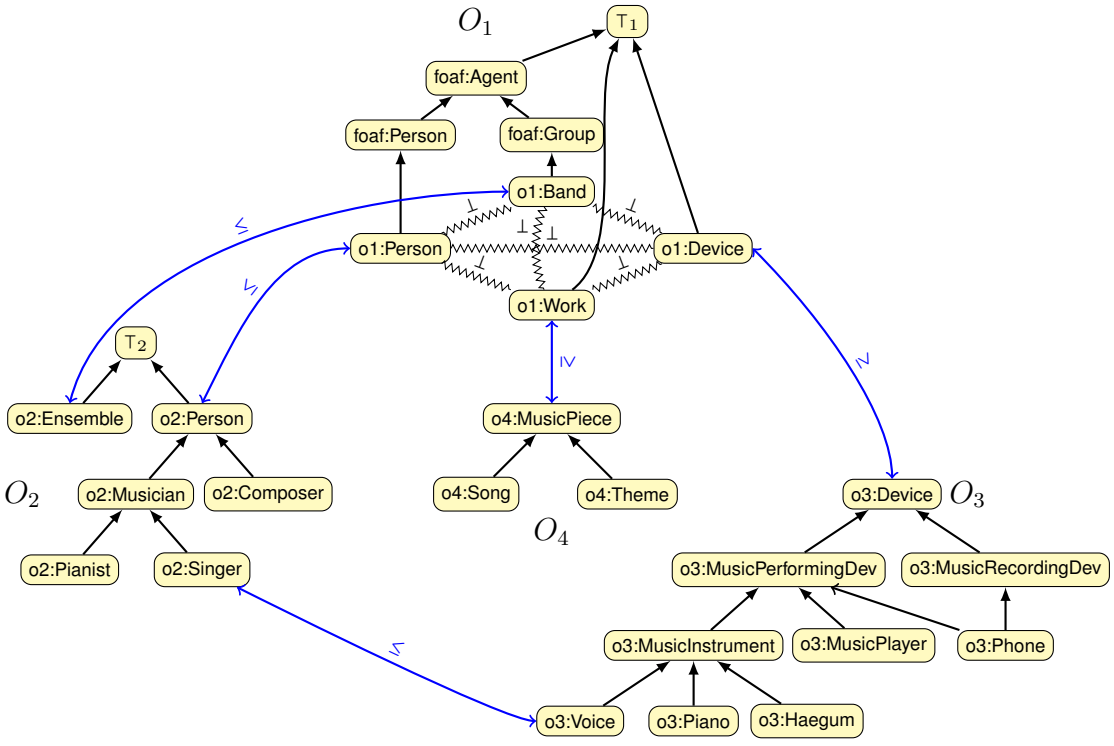


Figure 3.2. A network of ontologies N_1 made of four ontologies (O_1, O_2, O_3, and O_4) and four alignments ($A_{1,2}$, $A_{1,3}$, $A_{1,4}$, and $A_{2,3}$).

Models of networks of ontologies extend models of alignments. They select compatible models for each ontology in the network. Compatibility consists of satisfying all the alignments of the network.

Definition 3.7 (Models of networks of ontologies). *Given a network of ontologies $\langle \Omega, \Lambda \rangle$, a model of $\langle \Omega, \Lambda \rangle$ is a family of models $\{m_O\}_{O \in \Omega}$ indexed by Ω, such that:*
– $\forall O \in \Omega$, $m_O \in \mathcal{M}(O)$, *and*
– $\forall A \in \Lambda(O, O')$, $m_O, m_{O'} \vDash_\Delta A$.

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.

Hence, a model for the network of ontologies of Figure 3.2 with Δ as defined in Definition 3.4, is $\langle \{m_1, m_2, m_3, m_4\}, \Delta \rangle$ built on any models m_1, m_2, m_3 and m_4 of ontology O_1, O_2, O_3 and O_4 such that $m_2(\text{o2:Ensemble}) \subseteq m_1(\text{o1:Band})$, $m_2(\text{o2:Person}) \subseteq m_1(\text{o1:Person})$, $m_2(\text{o2:Singer}) \subseteq m_3(\text{o3:Voice})$, $m_3(\text{o3:Device}) \subseteq m_1(\text{o1:Device})$ and $m_4(\text{o4:MusicPiece}) \subseteq m_1(\text{o1:Work})$.

3.3 Consistency, entailment, and closure

From the alignment semantics, it is possible to decide if a network of ontologies is consistent or if a statement or correspondence is a consequence of the network.

3.3.1 Global and local consistency

An ontology or an alignment are consistent if they have a model, inconsistent otherwise. Similarly, a network of ontologies is consistent if it has a model.

Definition 3.8 (Local consistency, global consistency). *A network of ontology $\langle \Omega, \Lambda \rangle$ is* locally inconsistent *if*
– *there exists $O \in \Omega$ such that O is inconsistent, or*
– *there exists $A \in \Lambda$ such that A is inconsistent.*
It is globally inconsistent if it is locally consistent and has no model. Otherwise, it is consistent.

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: it cannot have a model. A network may be locally consistent, i.e. all ontologies and alignments have models, but globally inconsistent if their models are not compatible, i.e. they will interpret the same entity differently.

The network of ontologies presented in Figure 3.2 is incoherent since the class o2:Singer *is necessary empty (it entails* o2:Singer $\sqsubseteq \bot$*), i.e. for any family of models $\{m_1, m_2, m_3, m_4\}$ of $N_1 = \langle \{O_1, O_2, O_3, O_4\}, \{A_{1,2}, A_{1,3}, A_{2,3}, A_{1,4}\} \rangle$, $m_2(\text{o2:Singer}) \subseteq \varnothing$. This is indeed the case, $O_2 \vDash$ o2:Singer $\sqsubseteq$ o2:Person, but as mentioned at the end of Section 3.2 (p.33) $m_2(\text{o2:Person}) \subseteq m_1(\text{o1:Person})$, so $m_2(\text{o2:Singer}) \subseteq m_1(\text{o1:Person})$. Moreover, $m_2(\text{o2:Singer}) \subseteq m_3(\text{o3:Voice})$, but $m_3(\text{o3:Voice}) \subseteq m_3(\text{o3:Device})$ and $m_3(\text{o3:Device}) \subseteq m_1(\text{o1:Device})$ hence $m_2(\text{o2:Singer}) \subseteq m_1(\text{o1:Device})$. Thus, $m_2(\text{o2:Singer}) \subseteq m_1(\text{o1:Person}) \cap m_1(\text{o1:Device}) = \varnothing$.*

Figure 5.3 (p.60) displays a subnetwork $N_2 = \langle \{O_1, O_2', O_3\}, \{A_{1,2}, 1_{1,3}, A_{2,3}\} \rangle$ of N_1 (Figure 3.2) to which O_2 is added one statement: $O_2' = O_2 \cup \{\text{d2:Nina} \sqsubseteq \text{o2:Singer}\}$. A model of such a network would be a family of models $\{m_1, m_2, m_3\}$ of $\{O_1, O_2', O_3\}$ satisfying the alignments $A_{1,2}$, $A_{1,3}$ and $A_{2,3}$. But, all models m_2 of O_2' must satisfy $m_2(\text{d2:Nina}) \subseteq m_2(\text{o2:Person})$ because $m_2(\text{d2:Nina}) \in m_2(\text{o2:Singer}) \subseteq m_2(\text{o2:Musician}) \subseteq m_2(\text{o2:Person})$. This makes that $m_2(\text{d2:Nina}) \in m_1(\text{o1:Person})$ due to Alignment $A_{1,2}$. Moreover, due to Alignment $A_{2,3}$, $m_2(\text{d2:Nina}) \in m_3(\text{o3:Voice})$ and $m_3(\text{o3:Voice}) \subseteq m_3(\text{o3:MusicInstrument}) \subseteq m_3(\text{o3:MusicPerformingDev}) \subseteq m_3(\text{o3:Device})$, thus $m_2(\text{d2:Nina}) \in m_3(\text{o3:Device})$. In addition, due to Alignment $A_{1,3}$, $m_3(\text{o3:Device}) \subseteq m_1(\text{o1:Device})$, which means that $m_2(\text{d2:Nina}) \in m_1(\text{o1:Device})$. However, all models m_1 of O_1 satisfy $m_1(\text{o1:Person}) \cap m_1(\text{o1:Device}) = \varnothing$, hence it is not possible to find three models m_1, m_2 and m_3 which satisfy all these constraints. Thus, there is no model for this network of ontologies.

This network is inconsistent in the reduced interpretation, though none of its ontologies nor alignments is inconsistent. This is what is called a global inconsistency, by opposition to local inconsistency.

In addition, taking any of the ontologies with only the alignments which involve them, e.g. for O'_2, $\langle\{O_1, O'_2, O_3\}, \{A_{1,2}, A_{2,3}\}\rangle$, *is a consistent network of ontologies.*

3.3.2 Correspondence and formula entailment

There are two notions of consequences called ω-consequence and α-consequence.

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 3.9 (ω**-consequence).** *An assertion* δ *is the* ω*-consequence of an ontology* O *in a network of ontologies* $\langle\Omega, \Lambda\rangle$ *if*

$$\forall \langle m_x\rangle_{x\in\Omega} \in \mathcal{M}(\langle\Omega, \Lambda\rangle), m_O \vDash_{OWL} \delta.$$

This is noted $\langle\Omega, \Lambda\rangle \vDash_\Delta \delta/O$. The set of ω-consequences for an ontology O is $Cn^\omega_{\Omega,\Lambda}(O)$. The ω-closure of a network of ontologies is its set of ω-consequences: the assertions which are satisfied in all models of the network of ontologies.

According to these definitions, $Cn^\omega(O) = Cn^\omega_{\langle\{O\},\emptyset\rangle}(O)$. 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.

The simple consequences of the ontology O_2 *are:*

$$Cn^\omega(O_2) = \left\{\begin{array}{c} \text{o2:Ensemble} \sqsubseteq \top_2,\ \text{o2:Person} \sqsubseteq \top_2,\ \text{o2:Composer} \sqsubseteq \text{o2:Person},\\ \text{o2:Musician} \sqsubseteq \text{o2:Person},\ \text{o2:Pianist} \sqsubseteq \text{o2:Musician},\ \text{o2:Singer} \sqsubseteq \text{o2:Musician},\\ \text{o2:Musician} \sqsubseteq \top_2,\ \text{o2:Composer} \sqsubseteq \top_2,\ \text{o2:Pianist} \sqsubseteq \top_2,\\ \text{o2:Singer} \sqsubseteq \top_2,\ \text{o2:Pianist} \sqsubseteq \text{o2:Person},\ \text{o2:Singer} \sqsubseteq \text{o2:Person} \end{array}\right\}$$

while within N_1 *of Figure 3.2, there are even more consequences:*

$$Cn^\omega_{N_1}(O_2) = \left\{\begin{array}{c} \text{o2:Ensemble} \sqsubseteq \top_2,\ \text{o2:Person} \sqsubseteq \top_2,\ \text{o2:Composer} \sqsubseteq \text{o2:Person},\\ \text{o2:Musician} \sqsubseteq \text{o2:Person},\ \text{o2:Pianist} \sqsubseteq \text{o2:Musician},\ \text{o2:Singer} \sqsubseteq \text{o2:Musician},\\ \text{o2:Musician} \sqsubseteq \top_2,\ \text{o2:Composer} \sqsubseteq \top_2,\ \text{o2:Pianist} \sqsubseteq \top_2,\ \text{o2:Singer} \sqsubseteq \top_2,\\ \text{o2:Pianist} \sqsubseteq \text{o2:Person},\ \text{o2:Singer} \sqsubseteq \text{o2:Person},\ \text{o2:Ensemble} \perp \text{o2:Person},\\ \text{o2:Ensemble} \perp \text{o2:Musician},\ \text{o2:Ensemble} \perp \text{o2:Composer},\ \text{o2:Ensemble} \perp \text{o2:Singer},\\ \text{o2:Ensemble} \perp \text{o2:Pianist},\ \text{o2:Singer} \perp \text{o2:Person},\ \text{o2:Singer} \perp \text{o2:Musician},\\ \text{o2:Singer} \perp \text{o2:Singer} \end{array}\right\}$$

Similarly, α-consequences are correspondences which are consequences of networks of ontologies [Euzenat 2015].

Definition 3.10 (α**-consequence).** *A correspondence* μ *between two ontologies* O *and* O' *is an* α*-consequence of a network of ontologies* $\langle\Omega, \Lambda\rangle$ *iff*

$$\forall \langle m_x\rangle_{x\in\Omega} \in \mathcal{M}(\langle\Omega, \Lambda\rangle), m_O, m_{O'} \vDash_\Delta \mu$$

This is noted $\langle\Omega, \Lambda\rangle \vDash_\Delta \mu$. 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.

The closure of the alignment $A_{1,2} = \{\text{o1:Band} \geq \text{o2:Ensemble}, \text{o1:Person} \geq \text{o2:Person}\}$, *with respect to the two ontologies it is related to would be:*

$$Cn^{\alpha}_{\langle\{O_1,O_2\},\{A_{1,2}\}\rangle}(A_{1,2}) = \left\{ \begin{array}{c} \text{o1:Band} \geq \text{o2:Ensemble}, \text{o1:Person} \geq \text{o2:Person}, \text{foaf:Group} \geq \text{o2:Ensemble}, \\ \text{foaf:Agent} \geq \text{o2:Ensemble}, \top_1 \geq \text{o2:Ensemble}, \text{foaf:Person} \geq \text{o2:Person}, \\ \text{foaf:Agent} \geq \text{o2:Person}, \top_1 \geq \text{o2:Person}, \text{o1:Person} \geq \text{o2:Composer}, \\ \text{foaf:Person} \geq \text{o2:Composer}, \text{foaf:Agent} \geq \text{o2:Composer}, \top_1 \geq \text{o2:Composer}, \\ \text{o1:Person} \geq \text{o2:Musician}, \text{foaf:Person} \geq \text{o2:Musician}, \text{foaf:Agent} \geq \text{o2:Musician}, \\ \top_1 \geq \text{o2:Musician}, \text{o1:Person} \geq \text{o2:Pianist}, \text{foaf:Person} \geq \text{o2:Pianist}, \\ \text{foaf:Agent} \geq \text{o2:Pianist}, \top_1 \geq \text{o2:Pianist}, \text{o1:Person} \geq \text{o2:Singer}, \\ \text{foaf:Person} \geq \text{o2:Singer}, \text{foaf:Agent} \geq \text{o2:Singer}, \top_1 \leq \text{o2:Singer}, \\ \text{o1:Device} \bot \text{o2:Person}, \text{o1:Band} \bot \text{o2:Person}, \text{o1:Work} \bot \text{o2:Person}, \\ \text{o1:Device} \bot \text{o2:Composer}, \text{o1:Band} \bot \text{o2:Composer}, \text{o1:Work} \bot \text{o2:Composer}, \\ \text{o1:Device} \bot \text{o2:Musician}, \text{o1:Band} \bot \text{o2:Musician}, \text{o1:Work} \bot \text{o2:Musician}, \\ \text{o1:Device} \bot \text{o2:Pianist}, \text{o1:Band} \bot \text{o2:Pianist}, \text{o1:Work} \bot \text{o2:Pianist}, \\ \text{o1:Device} \bot \text{o2:Singer}, \text{o1:Band} \bot \text{o2:Singer}, \text{o1:Work} \bot \text{o2:Singer} \end{array} \right\}$$

In comparison, its closure within the network of ontology N_1 of Figure 3.2 would only add two correspondences:

$$Cn^{\alpha}_{N_1}(A_{1,2}) = Cn^{\alpha}_{\langle\{O_1,O_2\},\{A_{1,2}\}\rangle}(A_{1,2}) \cup \{\text{o1:Device} \geq \text{o2:Singer}, \text{o1:Person} \bot \text{o2:Singer}\}$$

It is noteworthy that, if the alignment $A'_{1,2}$ is empty, this still generates correspondences between O_1 and O_2:

$$Cn^{\alpha}_{\langle\{O_1,O_2,O_3,O_4\},\{A'_{1,2},A_{1,3},A_{1,4},A_{2,3}\}\rangle}(A'_{1,2}) = \left\{ \begin{array}{c} \text{o1:Device} \geq \text{o2:Singer}, \top_1 \geq \text{o2:Singer}, \text{o1:Band} \bot \text{o2:Singer}, \\ \text{o1:Work} \bot \text{o2:Singer}, \text{o1:Person} \bot \text{o2:Singer} \end{array} \right\}$$

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.

Finally, the closure function of a network of ontology is

$$Cn(\langle\Omega,\Lambda\rangle) = \langle\{Cn^{\omega}_{\Omega,\Lambda}(O)\}_{O\in\Omega}, Cn^{\alpha}_{\Omega,\Lambda}(\Lambda(O,O'))_{O,O'\in\Omega}\rangle.$$

The closure of a network of ontologies may introduce non empty alignments between ontologies which were not previously connected or empty. For instance, Figure 3.3 shows (in dashed) new correspondences $\langle$o2:Person, $\bot$, o4:MusicPiece$\rangle$ and $\langle$o4:MusicPiece, $\bot$, o3:Device$\rangle$ between previously non aligned ontologies O_2 and O_4 and O_3 and O_4, a new correspondence $\langle$o2:Singer, $\leq$, o3:MusicPerformingDev$\rangle$ within existing alignment $A_{2,3}$ and an additional statement o2:Ensemble owl:disjointWith o2:Person which is not entailed by O_2 alone. They are all entailed by the N_1 network, i.e. they hold in all models of the network.

3.4 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, HALPERN, MOSES, and VARDI 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 with respect to local knowledge (§3.4). 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 $\forall A \in \Lambda(O,O'), X \models_{\Delta} A$):

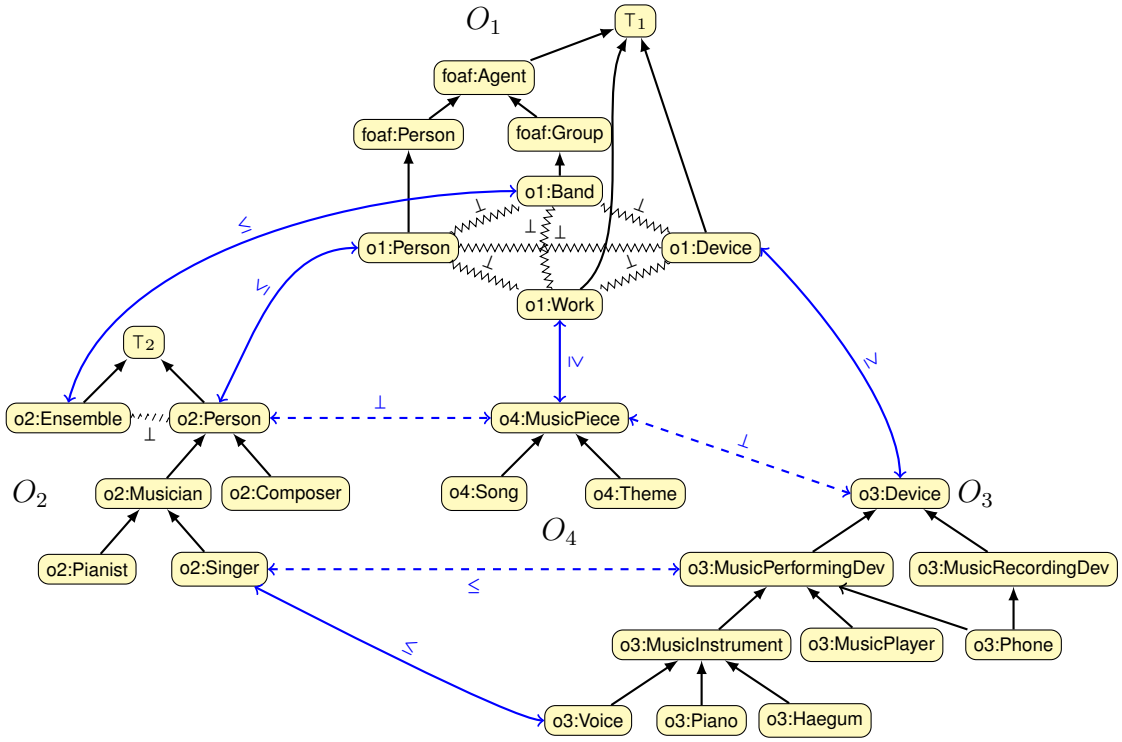


Figure 3.3. Alignment inference in the network of ontologies N_1 of Figure 3.2. The network now contains correspondences between O_1 and O_4 and O_4 and O_3 which were not aligned before. It also contains additional correspondences, e.g. o2:Singer $\leq$ o3:MusicPerformingDev, in $A_{2,3}$. O_2 contains new statements, e.g. o2:Ensemble$\perp$o2:Person, entailed by the network, but not the ontology alone.

$$\mathcal{M}^{0}_{\Omega,\Lambda}(O) = \mathcal{M}(O)$$
$$\mathcal{M}^{\exists}_{\Omega,\Lambda}(O) = \{m \in \mathcal{M}(O); \forall O' \in \Omega, \exists m' \in \mathcal{M}(O'); m, m' \vDash_{\Delta} \lambda(O, O')\}$$
$$\mathcal{M}^{\exists *}_{\Omega,\Lambda}(O) = \{m \in \mathcal{M}(O); \forall O' \in \Omega, \exists m' \in \mathcal{M}^{\exists *}_{\Omega,\Lambda}(O'); m, m' \vDash_{\Delta} \lambda(O, O')\}$$
$$\mathcal{M}^{\uparrow\exists}_{\Omega,\Lambda}(O) = \{m \in \mathcal{M}(O); \exists \overrightarrow{m} \in \mathcal{M}(\langle \Omega, \Lambda \rangle); \overrightarrow{m}_O = m\}$$
$$\mathcal{M}^{\forall *}_{\Omega,\Lambda}(O) = \{m \in \mathcal{M}(O); \forall O' \in \Omega, \forall m' \in \mathcal{M}^{\forall *}_{\Omega,\Lambda}(O'); m, m' \vDash_{\Delta} \lambda(O, O')\}$$
$$\mathcal{M}^{\forall}_{\Omega,\Lambda}(O) = \{m \in \mathcal{M}(O); \forall O' \in \Omega, \forall m' \in \mathcal{M}(O'); m, m' \vDash_{\Delta} \lambda(O, O')\}$$
$$\mathcal{M}^{\uparrow\forall}_{\Omega,\Lambda}(O) = \{m \in \mathcal{M}(O); \forall \overrightarrow{m} \in \mathcal{M}(\langle \Omega, \Lambda \rangle); \overrightarrow{m}_O = m\}$$

These approaches have been ordered from the most optimistic to the most 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 $\overrightarrow{m}_{O'}$ 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.

One can be even more restrictive by considering only a subset of the possible models of each ontology.

Conclusion

In this chapter, we built on the semantics for data and ontologies, to provide:

1. A language for aligning ontologies: $A \ni eRe'$ (correspondence)
2. A simple semantics for alignments: $A \models_\Delta eRe'$, $A \models_\Delta A'$, $A, O, O' \models_\Delta \delta$
3. A language for expressing networks of ontologies $\langle \Omega, \Lambda \rangle$
4. A semantics for such networks: $\langle \Omega, \Lambda \rangle \models_\Delta eRe'$, $\langle \Omega, \Lambda \rangle \models_\Delta \delta$

This defines what is entailed by a network of ontologies and their consistency even if ontologies are expressed with heterogeneous vocabularies.

For an agent, participating in a network, this restricts the models of its knowledge, hence provides more information (consequences).

This will be put to work in the next chapter to answer queries.

Quiz

– What does it mean for a correspondence to be the consequence of an alignment?
– Can a network of ontologies entail a correspondence between two ontologies between which there is no alignment?
– If a correspondence is entailed by an alignment, is it also entailed by any network of ontologies containing this alignment (and the two ontologies it connects)?
– Can adding a correspondence to an alignment have an effect on the models of related ontologies, and if yes which one?
– Is it possible that a network of ontologies be inconsistent?

References

[EUZENAT and SHVAIKO 2013] Jérôme Euzenat, Pavel Shvaiko, Ontology matching, Springer Verlag, Heidelberg (DE), 2007; 2nde edition, 2013

[EUZENAT 2015] Jérôme Euzenat, Revision in networks of ontologies, *Artificial intelligence* 228:195-216, 2015

The main support provided up-to-now for expressing ontology alignments is our own Alignment API [DAVID, EUZENAT, SCHARFFE, and TROJAHN 2011].

4

Distributed queries

IN WHICH WE CROSS THESE BRIDGES, EXPLOITING NETWORKS OF ONTOLOGIES AND LINKED DATA, IN ORDER TO BRING BACK ANSWERS TO QUERIES — THE RELATIONSHIP BETWEEN QUERY EVALUATION AND SEMANTICS BECOMES CLEAR.

One way to take advantage of knowledge is to answer queries. Such queries can be answered directly against the data graph known by the agent exploiting its ontology. However, it may be more informative to relay this query to other agents. In case they use heterogeneous ontologies, alignments of a network of ontologies may also be used for translating queries.

For that purpose, a query language is needed. I will concentrate here on a small subset of SPARQL. Although only one language is considered, I will discuss several ways to interpret, or more precisely evaluate, these queries when putting them into context.

Query answers must be defined with respect to the semantics of the languages used for expressing knowledge. The most basic query evaluation, simply match the query with the RDF graph and extract answers. This corresponds to the semantics of entailment. However, the same query may be evaluated in the context of an ontology, against several graphs at once or within a network of ontology. In each case, the answers will be provided with respect to the semantics of these components.

This is covered in another form in [EUZENAT 2007, Chapter 4 and 5].

4.1 SPARQL queries

I define in the following subsections the syntax of a small subset of the SPARQL language for expressing queries. This subset is simply that of conjunctive queries. It is sufficient to our purpose since it will allow us to provide different semantics to such queries in different, and in particular distributed, contexts.

For a complete description of SPARQL, the reader is referred to the SPARQL specification [PRUD'HOMMEAUX and SEABORNE 2008] or to [PÉREZ, ARENAS, and GUTIERREZ 2009; POLLERES 2007] for its formal semantics. Unless stated otherwise, I 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].

The heart of SPARQL queries is graph patterns [PRUD'HOMMEAUX and SEABORNE 2008]. Conjunctive queries simply use basic graph patterns: a set of triple patterns (or a GRDF graph).

Definition 4.1 (Basic graph pattern). *A basic graph pattern is an RDF graph in which some blank nodes are replaced by variables (noted $?x$).*

In the following, I treat blank nodes in RDF graphs simply as constants (as if they were IRIs) as done in the official specification of SPARQL without considering their existential semantics.

A conjunctive query is simply the projection of some variables of a basic graph pattern.

Definition 4.2 (Conjunctive query). *Given a basic graph pattern q, a tuple $\overrightarrow{x}$ of variables in q, and a reference u to a graph, a conjunctive query has the form:*

$$\text{SELECT } \vec{x} \text{ FROM } u \text{ WHERE } q$$

such a query will also be noted $q[\vec{x}]$.

The query q_1 *(Listing 4.1) asks for all pairs of resources related by the* mo:member *relation, that is all members of a musical ensemble:*

```
SELECT ?x, ?y WHERE { ?x mo:member ?y . }
```

Listing 4.1. Simple query.

Although, sequences of variables are considered, they can be thought of as sets. $\vec{x} \cap \vec{x}'$ *identifies variables common to two queries and* $\vec{x} \setminus \vec{x}'$ *those specific to one query.*

Query q_2 *(Listing 4.2) retrieves all titles of music pieces composed by someone named "Thelonius":*

```
SELECT ?title
WHERE {
  ?t1   mo:composer    ?g3 .
  ?g3   foaf:name      "Thelonious" .
  ?t1   dc:title       ?title .
}
```

Listing 4.2. Join query.

Query q_3 *(Listing 4.3) retrieves the name of a member of a band, and the name of that band, such that a member of the band has a relation with a piece of music that the band has performed:*

```
SELECT ?bnm, ?t1
WHERE {
  ?g3   foaf:name      "Thelonious" .
  ?b1   mo:member      ?g3 .
  ?g3   ?rel           ?t1 .
  ?g1   foaf:name      ?bnm .
  ?b1   mo:member      ?g1 .
  ?b1   mo:performed   ?t1 .
}
```

Listing 4.3. Graph query.

The query q_3 *graph pattern corresponds to the GRDF graph of Figure 2.2 (p. 10) with blank nodes turned into variables.*

Intuitively, an answer to a SELECT query is an assignment of the variables in $\vec{B}$ by terms of the RDF graph G such that, under these assignments, P is entailed by the graph identified by u.

Beside conjunctive queries, SPARQL provides other query forms.

Definition 4.3 (SPARQL query forms). *Given a SPARQL graph pattern* P*, a tuple* $\vec{x}$ *of variables in* P*, a IRI* u *and a basic graph pattern* Q*,*

$$\text{ASK FROM } u \text{ WHERE } P$$
$$\text{SELECT } \vec{x} \text{ FROM } u \text{ WHERE } P$$
$$\text{CONSTRUCT } Q \text{ FROM } u \text{ WHERE } P$$

are SPARQL queries.

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 sample query of Listing 4.4:

```
ASK
WHERE {
  ?piece       mo:composer    ?musician .
  ?ensemble    mo:member      ?musician .
  ?ensemble    mo:performed   ?piece .
}
```

Listing 4.4. ASK query.

returns TRUE *if the composer of a music piece has performed it within an ensemble,* FALSE *otherwise.*

The CONSTRUCT *query of Listing 4.5:*

```
CONSTRUCT {
   ?musician1     mo:playedWith    ?musician2  .
   ?musician2     mo:playedWith    ?musician1  .
   ?musician1     mo:playedIn      ?ensemble   .
   ?musician2     mo:playedIn      ?ensemble   .
} WHERE {
   ?ensemble      mo:member        ?musician1  .
   ?ensemble      mo:member        ?musician2  .
}
```

Listing 4.5. CONSTRUCT query.

constructs the RDF graph of the mo:playedWith *relation, i.e. all musicians having played in the same ensemble, and* mo:playedIn*, i.e. the converse of* mo:member *relation. The* mo:playedWith *relation is reflexive for those who have played in an ensemble.*

4.2 Genuine SPARQL semantics: query evaluation

In the following, I characterize query answering with SPARQL following [PÉREZ, ARENAS, and GUTIERREZ 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.

Definition 4.4 (Assignment (or map)). *A variable assignment is a map $\sigma : X \to V$, such that $X \subseteq \mathcal{V}$ and $V \subseteq \mathcal{T}$.*

If σ is an assignment, 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

$$\sigma|_X = \{\langle x, y\rangle \in \sigma |\ x \in X\}$$

If σ were a relational table, then $\sigma|_X = \pi_X(\sigma)$.

Assignment are extended from variables to basic graph patterns.

Definition 4.5 (Application of a map to a basic graph pattern). *Given a variable assignment $\sigma : X \to V$ and a basic graph pattern P, $\sigma(P)$, the application of σ to P is defined by:*

$$\begin{aligned}
\sigma(P) &= \{\sigma(t); t \in P\} && \text{if } P \text{ is an RDF graph}\\
\sigma(\langle s,p,o\rangle) &= \langle \sigma'(s), \sigma'(p), \sigma'(o)\rangle && \text{if } P \text{ is a triple}\\
\sigma'(x) &= \sigma(x) && \text{if } x \in dom(\sigma)\\
\sigma'(x) &= x && \text{otherwise.}
\end{aligned}$$

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 answers to a basic graph pattern query are those maps whose application warrants the entailment of the graph pattern by the queried graph.

The SPARQL semantics may be 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. In the case of SPARQL, this entailment relation is GRDF entailment.

Definition 4.6 (Answer to a SPARQL SELECT query [ALKHATEEB, BAGET, and EUZENAT 2009]). *Let $q[\vec{x}]$ be a conjunctive query and G be an RDF graph, then the set of answers to this query is*

$$\mathcal{A}(q[\vec{x}], G) = \{\sigma|_{\vec{x}} |\ G \models_{GRDF} \sigma(q)\}.$$

This definition is a semantic characterization of SPARQL answers.

The evaluation of the queries of Listing 4.1-4.3 against the graph G of Figure 2.1 provides the following answers:

$$\mathcal{A}(q_1[?x,?y],G) = \begin{Bmatrix} \{\langle ?x, \text{d2:g1}\rangle, \langle ?y, \text{d2:b1}\rangle\}, \\ \{\langle ?x, \text{d2:g3}\rangle, \langle ?y, \text{d2:b1}\rangle\}, \\ \{\langle ?x, \text{d2:g2}\rangle, \langle ?y, \text{d2:b2}\rangle\}, \\ \{\langle ?x, \text{d2:g3}\rangle, \langle ?y, \text{d2:b2}\rangle\} \end{Bmatrix}$$

$$\mathcal{A}(q_2[?title],G) = \{\{\langle ?title, \text{"Ruby, my dear"}\rangle\}\}$$

$$\mathcal{A}(q_3[?bnm,?t1],G) = \begin{Bmatrix} \{\langle ?bnm, \text{"John"}\rangle, \langle ?t1, \text{"Ruby, my dear"}\rangle\}, \\ \{\langle ?bnm, \text{"Thelonius"}\rangle, \langle ?t1, \text{"Ruby, my dear"}\rangle\}, \\ \{\langle ?bnm, \text{"Art"}\rangle, \langle ?t1, \text{"Ruby, my dear"}\rangle\} \end{Bmatrix}$$

Definition 4.7 (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*

$$\mathcal{A}^{ASK}(q[\overrightarrow{x}],G) = \exists\sigma;\ G \vDash_{GRDF} \sigma(q)$$

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

$$\mathcal{A}^{CONST}(q[\overrightarrow{x}],G,p) = \bigcup_{\sigma;\ G\vDash_{GRDF}\sigma(q)} \sigma(p)$$

Once given the semantics of a query, it is possible to compare two queries given their set of answers.

Definition 4.8 (Query containment). *A query $q[\overrightarrow{x}]$ is* contained *in a query $q'[\overrightarrow{x}]$ (noted $q[\overrightarrow{x}] \subseteq q'[\overrightarrow{x}]$), if* $\forall G,\ \mathcal{A}(q[\overrightarrow{x}],G) \subseteq \mathcal{A}(q'[\overrightarrow{x}],G)$.

Query containment is very useful for manipulating queries, e.g. replacing a query by another equivalent query for optimisation.

It can be shown that query $q_4[?bnm,?t1]$ of Listing 4.6:

```
SELECT ?bnm, ?t1
WHERE {
   ?g3   foaf:name        "Thelonious" .
   ?b1   mo:member        ?g3 .
   ?g3   ?rel             ?t1 .
   ?g3   foaf:name        ?bnm .
   ?b1   mo:performed     ?t1 .
}
```

Listing 4.6. Graph query.

is contained in query $q_3[?bnm,?t1]$, indeed, since $?g3$ is a member of the band, then it ("Thelonious"*) is always an answer. However, q_3 contains more answers such as* "John" *or* "Art" *in Figure 4.1.*

4.2.1 Computation

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

This problem has usually the same complexity as checking if an answer exists.

The complexity of checking RDF-entailment and GDRF-entailment is NP-complete [GUTIERREZ, HURTADO, and MENDELZON 2004]. This means that $\mathcal{A}$-ANSWER CHECKING when queries are reduced to basic graph patterns is NP-complete. For full SPARQL, the problem has been shown to be PSPACE-complete [PÉREZ, ARENAS, and GUTIERREZ 2009].

Figure 4.1 shows the projections for answering the queries q_1 of Listing 4.1 and q_3 of Listing 4.3 against Graph G of Figure 2.1.

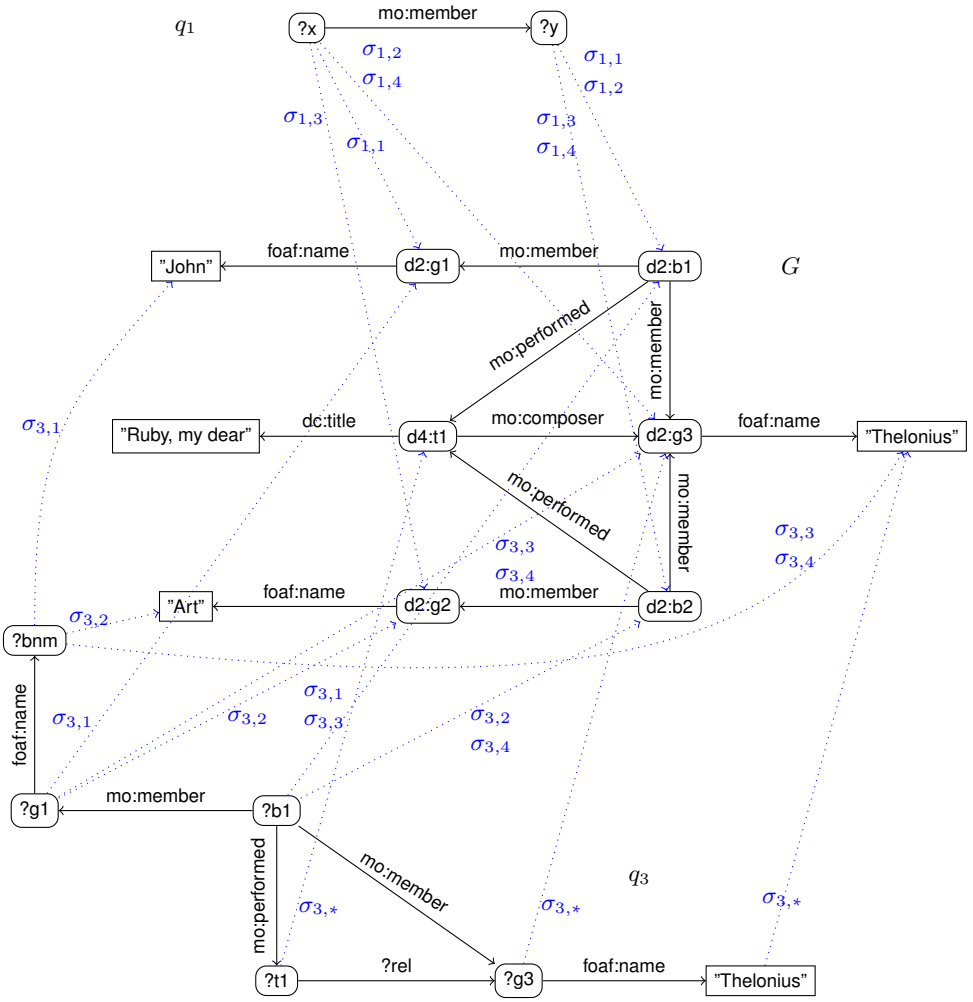


Figure 4.1. Conjunctive query evaluation through homomorphism.

4.3 Query evaluation modulo ontology

The genuine evaluation of SPARQL queries has only been considered with respect to a simple RDF graph (data). However, graphs may be expressed with respect to some ontology which should be used to interpret its meaning. Similarly, it may be useful to interpret the vocabulary of the query with respect to this ontology. This would allow the query engine to reason for answering queries. This may be useful, for instance, to answer all objects belonging to a class and not only those which are asserted to belong to that class.

For instance, one may want to interpret the query q of Listing 4.7:

```
SELECT ?x, ?y
WHERE {
  ?x        rdf:type          mo:Pianist .
  ?y        rdf:type          mo:JazzTheme .
  ?x        mo:hasComposed    ?y .
  ?e        mo:member         ?x .
  ?e        mo:performed      ?y .
}
```

Listing 4.7. Query modulo ontology.

against the graph G of Figure 4.2.

In principle, $\mathcal{A}(q[?x, ?y], G) = \varnothing$ because no object is asserted to be a mo:Pianist *nor a* mo:JazzTheme *and* mo:hasComposed *is not in the graph either. Hence, it is impossible to find a homomorphism between q and G.*

However, it is possible to evaluate this query against Ontology O of Figure 2.6 (p.21). On Figure 4.2, the edges added in dashed are OWL-entailed by $O \cup G$. They correspond to the inferences discussed p. 24, and thus comply with the semantics of OWL. This provides a homomorphism σ.

There are actually two homomorphisms: one mapping ?e *to* d2:b1 *and the other to* d2:b2. *However, since the projection does not retain* ?e, *this makes only one answer:* $\mathcal{A}^O(q[?x, ?y], G) = \{\langle$d2:g3, d4:t1$\rangle\}$.

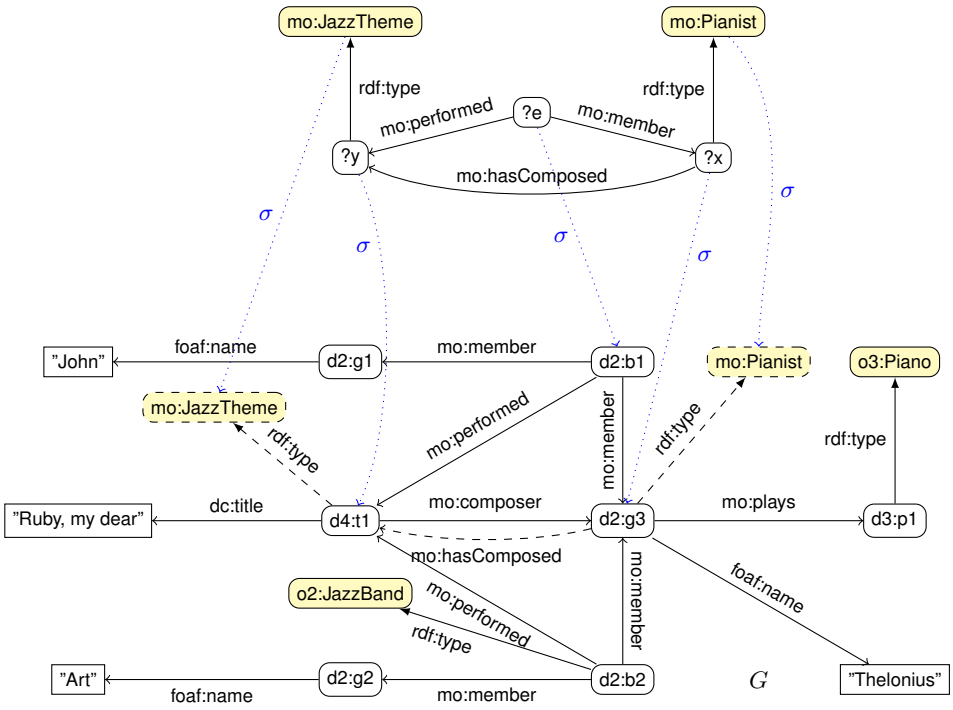


Figure 4.2. Query q has no answer in graph $G \cup O$. However, when G is complemented (in dashed) by assertions entailed by the ontology O of Listing 2.7, it is possible to provide a map σ.

What does it mean to take into account the ontology in which the graph is expressed? What does it mean to use this ontology to interpret the query?

Answering such queries by taking into account the ontology requires:

- to evaluate the query with the graph augmented by the ontology, and
- to evaluate it with OWL entailment.

Definition 4.9 (Answer to a SELECT query modulo ontology). *Let $q[\vec{x}]$ be a conjunctive query, G an RDF graph possibly containing ontological statements, then the set of answers to q modulo ontology in G is*

$$\mathcal{A}^O(q[\vec{x}], G) = \{\sigma|_{\vec{x}} \mid G \vDash_{OWL} \sigma(q)\}.$$

The $\vDash$ now has a different meaning because it considers the models of the RDF graph in the OWL semantics.

SPARQL 1.1 has introduced the concept of entailment regime [GLIMM and OGBUJI 2013] for defining this. It is slightly more sophisticated than what is presented here since it pays attention at limit cases, such as when the graph and the ontology are inconsistent.

4.4 Query evaluation over several data sources

This is quickly discussed in [EUZENAT 2007, Chapter 12].

Data and ontologies may be distributed in several sources. It is possible that some sources contain some answers to a query and other sources contain other answers. Answering in a more extensive way to queries requires to exploit all of these sources. It is thus necessary to provide the semantics of querying against several sources which are as many RDF graphs.

Consider the query q of Listing 4.8:

```
SELECT ?x, ?y
WHERE {
  ?x          rdf:type        foaf:Person .
  ?y          rdf:type        foaf:Group .
  ?y          foaf:member     ?x .
}
```

Listing 4.8. Query q to be dispatched.

expressed in Figure 4.3. This query can simply be sent to various sources: $O_1 \cup G_1$, $O_2 \cup S \cup G_2$, $O_3 \cup G_3$, and $O_4 \cup G_4$ yielding the following answers:

- $\mathcal{A}^O(q[?x,?y], G_1 \cup O_1) = \{\langle$d1:g7, d1:b4$\rangle, \langle$d1:g5, d1:b3$\rangle\}$
- $\mathcal{A}^O(q[?x,?y], G_2 \cup S \cup O_2) = \{\langle$d2:g1, d2:b1$\rangle, \langle$d2:g3, d2:b1$\rangle, \langle$d2:g2, d2:b2$\rangle, \langle$d2:g3, d2:b2$\rangle\}$
- $\mathcal{A}^O(q[?x,?y], G_3 \cup O_3) = \varnothing$
- $\mathcal{A}^O(q[?x,?y], G_4 \cup O_4) = \varnothing$

Hence, $\mathcal{A}^O_\cup(q[\overrightarrow{x}], \{G_1 \cup O_1, G_2 \cup S \cup O_2, G_3 \cup O_3, G_4 \cup O_4\}) = \mathcal{A}^O(q[?x,?y], G_1 \cup O_1) \cup \mathcal{A}^O(q[?x,?y], G_2 \cup S \cup O_2) \cup \mathcal{A}^O(q[?x,?y], G_3 \cup O_3) \cup \mathcal{A}^O(q[?x,?y], G_4 \cup O_4) = \{\langle$d2:g1, d2:b1$\rangle, \langle$d2:g3, d2:b1$\rangle, \langle$d2:g2, d2:b2$\rangle, \langle$d2:g3, d2:b2$\rangle, \langle$d1:g7, d1:b4$\rangle, \langle$d1:g5, d1:b3$\rangle\}$.

These graphs may be redundant, i.e. providing several times the same answer, contradictory, i.e. put together they have no model, or complementary, they only provide part of the answers.

The question is what is the meaning (what are the answers) of a query $q[\overrightarrow{x}]$ evaluated against several graphs $G_1, \ldots G_n$ (noted $\{G_i\}_{i\in I}$)? The problem is how to evaluate them so that they provide these answers. It can be simply solved by evaluating the query against all the data sources and collecting the answers. Figure 4.3 provide an example of such a situation.

Definition 4.10 (Simple distributed answers to a SELECT query modulo ontology). *Let $q[\overrightarrow{x}]$ be a conjunctive query, and $\{G_i\}_{i\in I}$ a set of RDF graphs, then the set of answers to this query is*

$$\mathcal{A}^O_\cup(q[\overrightarrow{x}], \{G_i\}_{i\in I}) = \bigcup_{i\in I} \mathcal{A}^O(q[\overrightarrow{x}], G_i).$$

It is possible that some answers can be computed from the union of the graph, but not from any of the sources independently. This is what could cause

$$\mathcal{A}^O_\cup(q[\overrightarrow{x}], \{G_i\}_{i\in I}) \subseteq \mathcal{A}^O(q[\overrightarrow{x}], \bigcup_{i\in I} G_i).$$

In order to answer queries in a truly distributed manner, it is necessary to be able to combine answers to subqueries. For this, it is first necessary to partition queries in sets of subsqueries.

Definition 4.11 (Query partition). *Given a query $q[\overrightarrow{x}]$ and a set of graphs $\{G_i\}_{i\in I}$, a partition of the query $q[\overrightarrow{x}]$ over $\{G_i\}_{i\in I}$ is a set of pairs $\{\langle q_j[\overrightarrow{x}_j], G_j\rangle\}_{j\in J}$ such that*

- *$J \subseteq I$,*
- *$q_j \subseteq q$ is a subgraph of q using only the vocabulary $\mathcal{V}_j$ of graph G_j, and*
- *each triple of q is in exactly one of the q_j,*
- *each q_j contains at least one triple,*
- *$\overrightarrow{x_j}$ is the set of variables appearing in q_j ($\overrightarrow{x_j}$ may contain variables not in $\overrightarrow{x}$).*

The set of query partitions of $q[\overrightarrow{x}]$ over $\{G_i\}_{i\in I}$ is $QPart(q[\overrightarrow{x}], \{G_i\}_{i\in I})$.

Hence a partition does not necessary use all the graphs, but it must cover the whole query.

The sets of partitions of a query may be very large: the number of possible J is exponential in $|I|$, the set of possible distribution of the triples of q is also exponential in the size of q. In the worst case, all data sources can answer the query. However, in practical cases it is possible that only a few partitions are applicable.

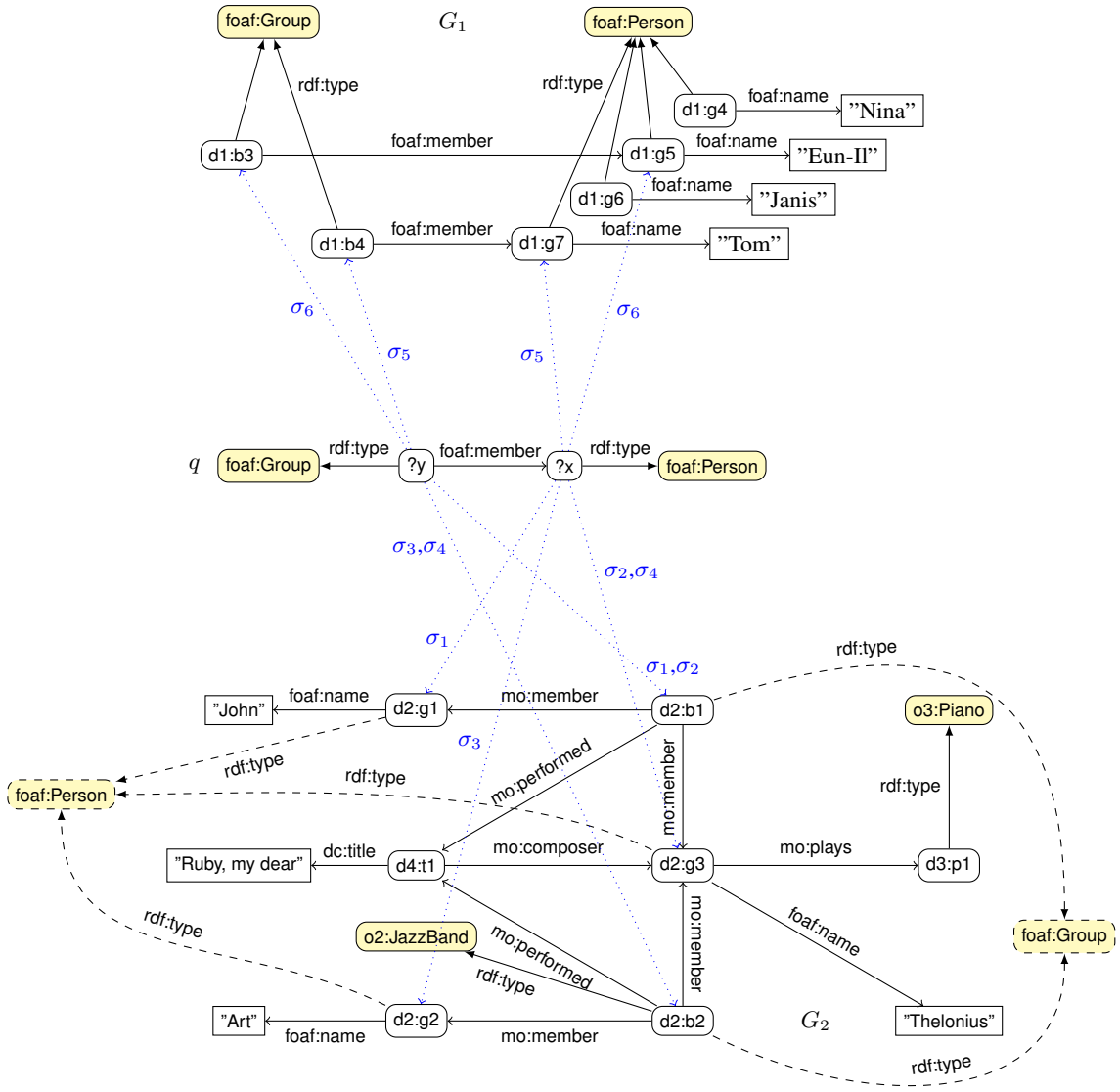


Figure 4.3. Simple distributed query evaluation modulo ontology. The query q is evaluated against graph $G_1 \cup O_1$ and $G_2 \cup S \cup O_2$ respectively returning the maps σ_1 to σ_6.

Figure 4.4 provides an example of how to evaluate relevant subqueries that are compatible and thus may be used to return a single answer. In this case the partition is made of the queries q_1 and q_2.

Consider the query q of Listing 4.9:

```
SELECT ?y, ?z
WHERE {
  ?x        rdf:type        foaf:Person .
  ?y        rdf:type        o2:Ensemble .
  ?z        rdf:type        o4:MusicPiece .
  ?y        foaf:member     ?x .
  ?z        dc:creator      ?x .
}
```

Listing 4.9. Query q to be partitioned.

expressed in Figure 4.4. If this query is dispatched as such to each data source as in the previous example, it would return no answer (there are no o2:Ensemble *in* $O_4 \cup G_4$*, nor any* o4:MusicPiece *in* $O_2 \cup S \cup G_2$*). It is*

however possible to partition it into two subqueries that can be understood by different data sources. Indeed:

q_1

```
SELECT ?x, ?y
WHERE {
  ?x    rdf:type        foaf:Person .
  ?y    rdf:type        o2:Ensemble .
  ?y    foaf:member     ?x .
}
```

Listing 4.10. Subquery q_1.

q_2

```
SELECT ?x, ?z
WHERE {
  ?x    rdf:type        foaf:Person .
  ?z    rdf:type        o4:MusicPiece .
  ?z    dc:creator      ?x .
}
```

Listing 4.11. Subquery q_2.

are two subqueries that can be dispatched to $O_1 \cup G_1$ *and* $O_2 \cup S \cup G_2$ *(for* q_1*) and* $O_4 \cup G_4$ *(for* q_2*). On the one hand,* $\mathcal{A}^O(q_1[?x,?y], O_1 \cup G_1) \cup \mathcal{A}^O(q_1[?x,?y], O_2 \cup S \cup G_2)$ *will return the same set of answers as the example of Figure 4.3. On the other hand,* $\mathcal{A}^O(q[?x,?z], O_4 \cup G_4) = \{\langle$d2:g4, d4:t3$\rangle, \langle$d2:g1, d4:t1$\rangle, \langle$d2:g6, d4:t4$\rangle, \langle$d2:g3, d4:t2$\rangle\}$.

The results of $\mathcal{A}^O(q[?y,?z], \{O_1 \cup G_1, O_2 \cup S \cup G_2, O_3 \cup G_3, O_4 \cup G_4\})$ *will be* $\pi_{?y,?z}(\mathcal{A}^O(q_1[?x,?y], O_1 \cup G_1) \cup \mathcal{A}^O(q_1[?x,?y], O_2 \cup S \cup G_2) \bowtie \mathcal{A}^O(q[?x,?z], O_4 \cup G_4))$*, i.e.* $\{\langle$d2:b1, d4:t1$\rangle, \langle$d2:b1, d4:t2$\rangle, \langle$d2:b2, d4:t2$\rangle\}$.

It is now possible to answer queries in a distributed way:

Definition 4.12 (Distributed answers to a SELECT query modulo ontology). *Let* $q[\overrightarrow{x}]$ *be a conjunctive query and* $\{G_i\}_{i\in I}$ *a set of RDF graphs, then the set of answers to this query is*

$$\mathcal{A}^O(q[\overrightarrow{x}], \{G_i\}_{i\in I}) = \bigcup_{\{\langle q_j[\overrightarrow{x}_j], G_j\rangle\}_{j\in J} \in QPart(q[\overrightarrow{x}], \{G_i\}_{i\in I})} \{\sigma|_{\overrightarrow{x}} \mid \sigma \in \bowtie_{j\in J} \mathcal{A}^O(q_j[\overrightarrow{x_j}], G_j)\}.$$

The natural join ($\bowtie$) is used to ensure that each variable has the same value in each combined answer.

This is a generalisation of the previous case.

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). In this case, the partition has to be provided by the query designer.

4.5 Query evaluation in networks of ontologies

As we have seen in Section 3.2, the data sources available may not be expressed with the same vocabulary, defined in the same ontology. This is an obstacle to distributed querying. However, ontologies related through alignments may offer the opportunity to translate queries from an ontology to another. In other words, alignments may be used as mediators between heterogeneous data sources. This is illustrated by Figure 4.5.

Networks of ontologies allow for connecting ontologies together in a meaningful way. Hence, they can help interpreting information from one ontology into another ontology.

This means that queries will be evaluated by using the alignments between the ontologies associated to the graphs. This leads to a different partition of a query.

Definition 4.13 (Query partition modulo alignments). *Given a query* $q[\overrightarrow{x}]$ *over an ontology* O*, a set of graphs* $\{G_i\}_{i\in I}$ *expressed in ontologies* $\{O_i\}_{i\in I}$*, and a set of alignments* $\{A_i\}_{i\in I}$ *between* O *and* $\{O_i\}_{i\in I}$*, a partition of the query* $q[\overrightarrow{x}]$ *is a set of pairs* $\{\langle q_j[\overrightarrow{x}_j], G_j\rangle\}_{j\in J}$ *such that*

- $J \subseteq I$,
- $q'_j \subseteq q$ *is a subgraph of* q *using only the vocabulary in the domain of* A_j*;*
- $q_j = A_j(q'_j)$ *is the transformation of* q'_j *according to alignment* A_j*.*
- *each triple of* q *is in exactly one of the* q'_j*,*
- *each* q_j *contains at least one triple,*
- $\overrightarrow{x_j}$ *is the set of variables appearing in* q_j *(*$\overrightarrow{x_j}$ *may contain variables not in* $\overrightarrow{x}$*).*

The set of query partitions of $q[\overrightarrow{x}]$ *over* $\{G_i\}_{i\in I}$ *modulo alignments* $\{A_i\}_{i\in I}$ *is* $QPart(q[\overrightarrow{x}], \{G_i\}_{i\in I}, \{A_i\}_{i\in I})$.

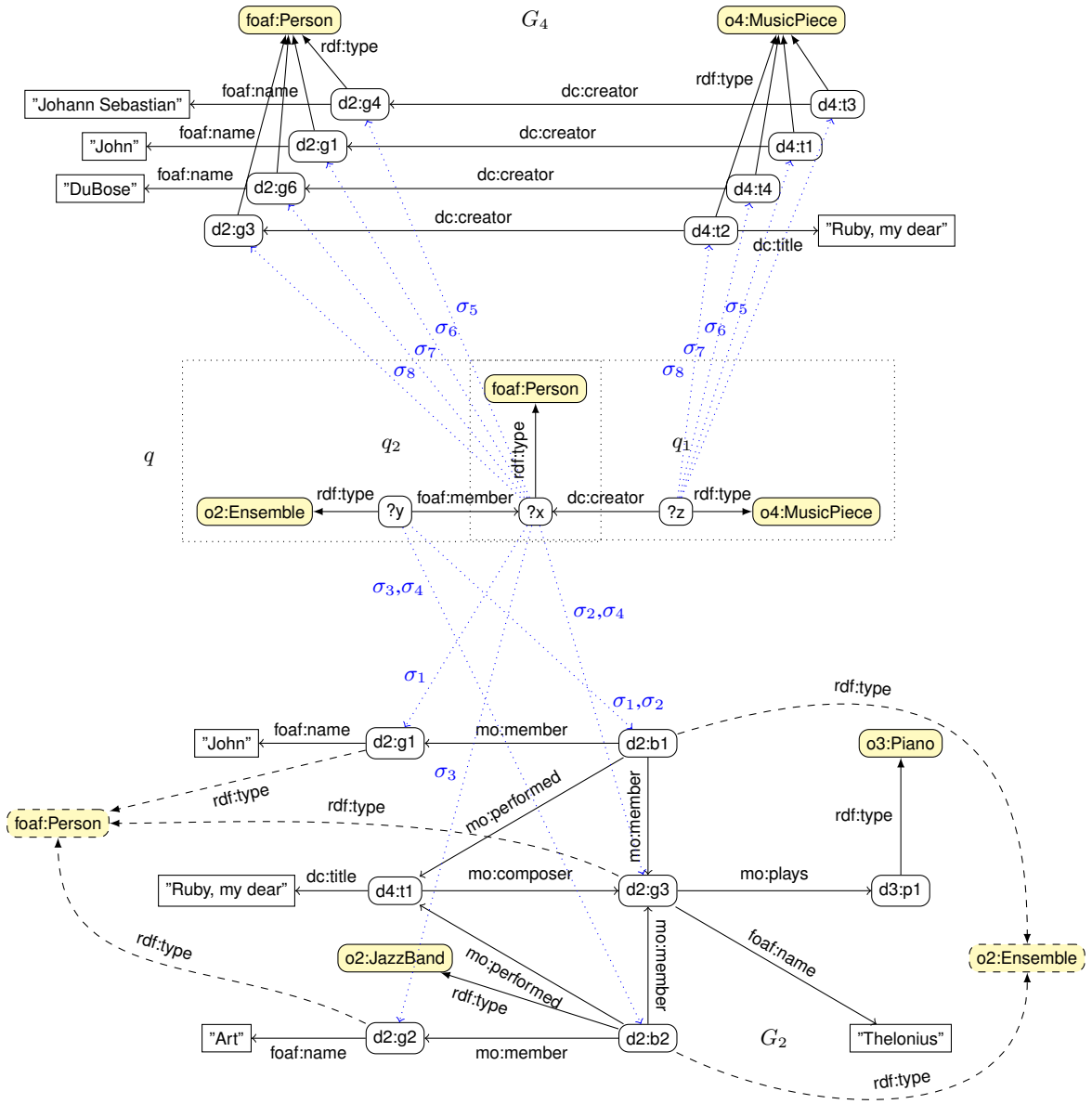


Figure 4.4. Distributed query modulo ontology. Only some parts of the query q are evaluated against graphs G_2 and G_4. The corresponding sets of answers are joined to provide an answer with respect to the initial query.

A particular alignment may be the identity alignment in case two sources use the same ontology.

Here we need a specific join operator which works with equalities as well as linked IRIs. Indeed, the natural join ($\bowtie$) considered in Definition 4.12 can be defined as:

$$\mathcal{A}(q[\vec{x}]) \bowtie \mathcal{A}(q'[\vec{x'}]) = \{\sigma|_{\vec{x}\cap\vec{x'}} \cup \sigma|_{\vec{x}\setminus\vec{x'}} \cup \sigma'|_{\vec{x'}\setminus\vec{x}} \mid \langle\sigma,\sigma'\rangle \in \mathcal{A}(q[\vec{x}]) \times \mathcal{A}(q'[\vec{x'}]) \wedge \sigma|_{\vec{x}\cap\vec{x'}} = \sigma'|_{\vec{x}\cap\vec{x'}}\}$$

as answers are variable assignments. If the data sources use different vocabulary, it is likely that individuals returned as answers are identified by different IRIs. This would prevent the join to occur. The alignments may be used in order to express that different IRIs denote the same individual through sameAs correspondences (=). In order to take advantage of these in the join, one has to consider a set of alignments Λ. The set of sameAs correspondences that it entails ($\Lambda \models_\Delta \langle o, =, o'\rangle$) forms an equivalence relation. The new join is expressed as:

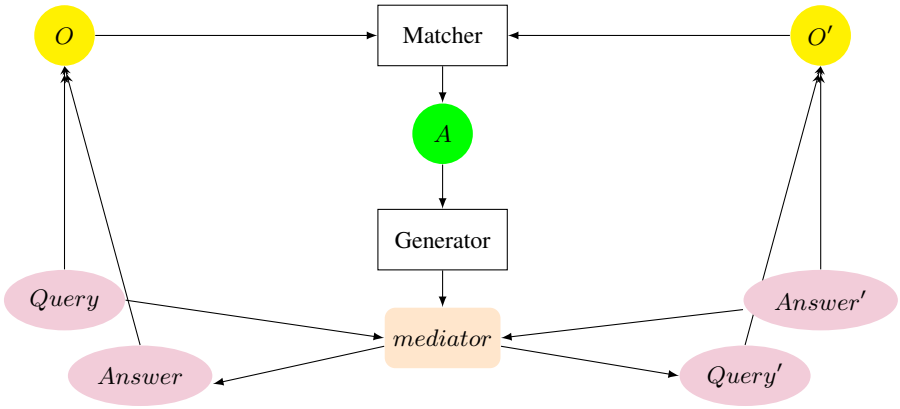


Figure 4.5. Query mediation (from [EUZENAT and SHVAIKO 2013]). From two matched ontologies O and O', resulting in alignment A, a *mediator* is generated. This allows the transformation of queries expressed with the entities of the first ontology into a query using the corresponding entities of a matched ontology and the translation back of the results from the second ontology to the first one.

$$\mathcal{A}(q[\vec{x}]) \bowtie^{\Lambda} \mathcal{A}(q'[\vec{x'}]) = \{\sigma|_{\vec{x}\cap\vec{x'}} \cup \sigma|_{\vec{x}\setminus\vec{x'}} \cup \sigma'|_{\vec{x'}\setminus\vec{x}} \mid \langle\sigma,\sigma'\rangle \in \mathcal{A}(q[\vec{x}]) \times \mathcal{A}(q'[\vec{x'}]) \\ \wedge \forall v \in \vec{x}\cap\vec{x'}, \Lambda \vDash_{\Delta} \langle\sigma(v), =, \sigma'(v)\rangle\}$$

noting that Λ will always satisfy $\langle o, =, o\rangle$ due to the semantics of alignments (Definition 3.4).

Then it is possible to answer the query as:

Definition 4.14 (Distributed answers to a SELECT query in a network of ontologies). *Let $q[\vec{x}]$ be a conjunctive query expressing in an ontology O, $\{G_i\}_{i\in I}$ be RDF graphs expressed in their respective ontologies $\{O_i\}_{i\in I}$, $\{A_i\}_{i\in I}$ be ontology alignments between O and $\{O_i\}_{i\in I}$, then the set of answers to this query is*

$$\mathcal{A}^{O}(q[\vec{x}], \{G_i\}_{i\in I}, \{A_i\}_{i\in I}) = \bigcup_{\{\langle q_j[\vec{x}_j], G_j\rangle\}_{j\in J}\in QPart(q[\vec{x}],\{G_i\}_{i\in I},\{A_i\}_{i\in I})} \{\sigma|_{\vec{x}} \mid \sigma \in \bowtie_{j\in J}^{\{A_i\}_{i\in I}} \mathcal{A}^{O}(q_j[\vec{x_j}], G_j)\}.$$

This is a further generalisation of the previous case.

How does this relate to:

$$\mathcal{A}^{\Omega,\Lambda}(q[\vec{x}], \{G_i\}_{i\in I}) = \{\sigma|_{\vec{x}} \mid \langle\Omega,\Lambda\rangle \vDash_{\Delta} \sigma(q)\}?$$

In the previous examples (Listing 4.8 and 4.9) it was possibe to evaluate the queries because the graphs were sharing common fragments of ontologies, namely FOAF and Dublin core. This is not always the case, like different people may understand different languages, data sources may be expressed in different ontologies. As discussed in Section 3.2, this can be supplemented by alignments within networks of ontologies. Consider the query q of Listing 4.12:

```
SELECT ?x, ?y
WHERE {
  ?x          rdf:type        foaf:Person .
  ?y          rdf:type        o1:Band .
  ?z          rdf:type        o1:Work .
  ?y          mo:member       ?x .
  ?z          mo:composer     ?x .
}
```

Listing 4.12. Query q to be partitioned through alignments.

expressed in Figure 4.6. If this query is dispatched as such to each data source as in for Listing 4.8 against several data sources (§4.4), it would return no answer (there are no o1:Band *nor* o1:Work *in any of $O_4 \cup G_4$ or $O_2 \cup G_2$). Consider two alignments ($A_{1,2}$ and $A_{1,4}$):*

$$A_{1,2} = \{\langle \text{o1:Band}, \geq, \text{o2:Ensemble}\rangle\} \qquad A_{1,4} = \{\langle \text{o1:Work}, \geq, \text{o4:MusicPiece}\rangle\}$$

and an alignment $A_{2,4} = \{\langle$d2:b3, =, d4:b3$\rangle\}$*, which may be called a link set, it is possible to partition* q *into two subqueries that can be translated with the alignments and understood by the different data sources. Indeed:*

```
SELECT ?x, ?y
WHERE {
  ?x    rdf:type        foaf:Person .
  ?y    rdf:type        o1:Band .
  ?y    foaf:member     ?x .
}
```

Listing 4.13. Subquery q_2.

```
SELECT ?x, ?z
WHERE {
  ?x    rdf:type        foaf:Person .
  ?z    rdf:type        o1:Work .
  ?z    dc:creator      ?x .
}
```

Listing 4.14. Subquery q_4.

may be translated as:

```
SELECT ?x, ?y
WHERE {
  ?x    rdf:type        foaf:Person .
  ?y    rdf:type        o2:Ensemble .
  ?y    foaf:member     ?x .
}
```

Listing 4.15. Translated subquery $A_{1,2}(q_2)$.

```
SELECT ?x, ?z
WHERE {
  ?x    rdf:type        foaf:Person .
  ?z    rdf:type        o4:MusicPiece .
  ?z    dc:creator      ?x .
}
```

Listing 4.16. Translated subquery $A_{1,4}(q_4)$.

The translated subqueries can be dispatched to $O_2 \cup G_2$ *(for* $A_{1,2}(q_2)$*) and* $O_4 \cup G_4$ *(for* $A_{1,4}(q_4)$*). On the one hand,* $\mathcal{A}^O(A_{1,2}(q_2)[?x, ?y], O_2 \cup G_2)$ *will return only two answers* $\{\langle$d2:g2, d2:b2$\rangle, \langle$d2:g3, d2:b2$\rangle\}$*. On the other hand,* $\mathcal{A}^O(A_{1,4}(q_4)[?x, ?z], O_4 \cup G_4)$ *will return the same set of answers as Listing 4.9 evaluated against several data sources (§4.4), i.e.* $\{\langle$d4:g4, d4:t3$\rangle, \langle$d4:g1, d4:t1$\rangle, \langle$d4:g6, d4:t4$\rangle, \langle$d4:g3, d4:t2$\rangle\}$.

The results of $\mathcal{A}^O(q[?y, ?z], \{O_2 \cup G_2, O_4 \cup G_4\}, \{A_{1,2}, A_{1,4}\})$ *will be* $\pi_{?y,?z}(\mathcal{A}^O(A_{1,2}(q_2)[?x, ?y], O_2 \cup G_2) \bowtie \mathcal{A}^O(A_{1,4}(q_4)[?x, ?z], O_4 \cup G_4)) = \{\langle$d2:g3, d4:t2$\rangle\}$ *(or, equivalently with respect to* $A_{2,4}$*,* $\{\langle$d4:g3, d4:t2$\rangle\}$*).*

4.6 Computation

Implementing distributed query evaluation is a difficult task with many issues to solve. It may be implemented by importing all data into a single warehouse (materialisation) or, closer to the semantics, by rewriting queries.

What has been presented here is again *only* the semantics. The way alignments are used may be considered as local-as-view (LAV) as soon as the query classes or properties are mapped through $\leq$. However, the query partitioning schemes makes it global-as-view (GAV) automatically as they map full queries to these single classes and properties.

However, the answer provided by a network of ontologies is a view from above: everything is known about the data and knowledge of everyone. This may not be the case in practice.

A distributed approach, one the other hand, may be less complete, but provides correct answers based on available information. Technically, distributed approaches may seem more complex and in particular, the number of possible partitions is daunting. This is the reason why, distributed query evaluators are far more complex to implement: they will not explore the partition set entirely but use traversal strategies for that purpose, they will not evaluate each partitions but select the most promising, they will not evaluate queries independently but instead build query plans, based on cost models, aiming at balancing performances and completeness. This would involve identifying sources, selecting sources (a priori or at evaluation time), partitioning queries, and scheduling and monitoring evaluation.

Note that we have only presented here a very simple fragment of SPARQL based on conjunctive queries. In practice, queries may have operators (disjunctions, outer join) that make them more elaborate. However, this also provides a more flexible query algebra that can be used to rewrite queries *in the system*. This is because the query optimiser is 'knowledgeable' of the SPARQL semantics, e.g. through query subsumption, that it can manipulate queries so that it evaluates more efficient queries that will return correct results.

Another issue, already mentioned, is the consistency of the network of ontologies. In case of inconsistency, any answers is correct with respect to the semantics. To achieve this result, it is not worth evaluating queries. A consistent subnetwork, may provide interesting answers...

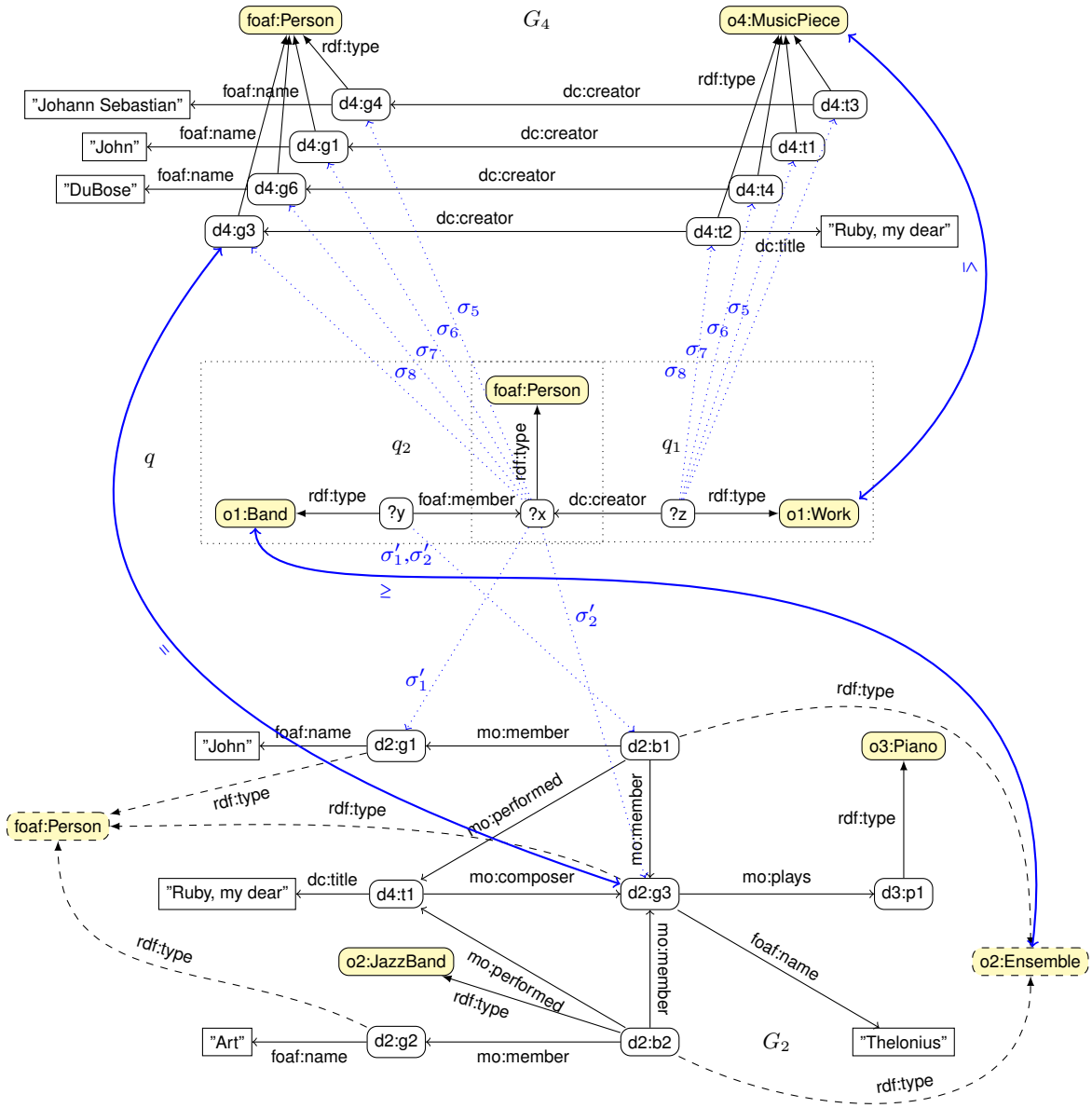


Figure 4.6. Query evaluation in a network of ontologies. The query q and the two graphs G_2 and G_4 do not use the same ontologies but alignments ($A_{1,2}$ and $A_{1,4}$) can translate the query vocabulary in that of the graphs. The individual correspondence in $A_{2,4}$ matches d4:g3 and d2:g3.

Conclusion

We provided a language for expressing queries ($q[\vec{x}]$) and different semantics for queries specifying their answers ($\mathcal{A}(q[\vec{x}], G)$), modulo ontologies ($\mathcal{A}^O(q[\vec{x}], G)$) and in a distributed ($\mathcal{A}^O(q[\vec{x}], \{G_i\}_{i\in I})$ and heterogeneous context ($\mathcal{A}^{\Omega,A}(q[\vec{x}], \{G_i\}_{i\in I})$. This allows to retrieve information, not simply in a single data set, but by taking advantage of data and knowledge distributed in different heterogeneous sources. The query semantics do not bring new semantics: they use the defined semantics for graphs, ontologies, alignments and networks of ontologies.

In the end, it can be argued that:

$$
\begin{aligned}
& \mathcal{A}(q[\overrightarrow{x}], G) \\
\subseteq\ & \mathcal{A}^{O}(q[\overrightarrow{x}], G) \\
\subseteq\ & \mathcal{A}^{O}_{\cup}(q[\overrightarrow{x}], \{G_i\}_{i\in I}) && \text{if } G \in \{G_i\}_{i\in I} \\
\subseteq\ & \mathcal{A}^{O}(q[\overrightarrow{x}], \{G_i\}_{i\in I}) \\
\subseteq\ & \mathcal{A}^{O}(q[\overrightarrow{x}], \{G_i\}_{i\in I}, \{A_i\}_{i\in I}) \\
\subseteq\ & \mathcal{A}^{\Omega,A}(q[\overrightarrow{x}], \{G_i\}_{i\in I})
\end{aligned}
$$

Said otherwise, the more information (ontologies, sources, alignments, networks) is exploited, the more answers are given to queries or the more complete this set of answers is with respect to those given by a network of ontologies.

Quiz

- For what is it useful to query different sources?
- What should be the answer to a query if the graph to which it is applied is inconsistent? Why?
- Could a query be inconsistent? If it is possible, can it have answers?
- Should querying with respect to an ontology return more answers than without it? Why?

References

[ABITEBOUL, MANOLESCU, RIGAUX, ROUSSET, and SENELLART 2011] Serge Abiteboul, Ioana Manolescu, Philippe Rigaux, Marie-Christine Rousset, Pierre Senellart, Web Data Management, Cambridge university press (UK), 2011

[ADJIMAN, CHATALIC, GOASDOUÉ, ROUSSET, and SIMON 2006] Philippe Adjiman, Philippe Chatalic, Francois Goasdoué, Marie-Christine Rousset, Laurent Simon, Distributed Reasoning in a Peer-to-Peer Setting: Application to the Semantic Web, *Journal of artificial intelligence research* Volume 25. 2006.

Queries and their semantics may be defined along the W3C SPARQL recommendation [HARRIS and SEABORNE 2013].

5

Belief revision

IN WHICH WE FACE INCONSISTENCY, OR THE ABSENCE OF MODEL — IT SPREADS TO THE WHOLE NETWORK OF ONTOLOGIES AND CONTAMINATES QUERY RESULTS — ISOLATING MAXIMAL CONSISTENT SUBNETWORKS AND REPAIRING INCONSISTENCY REVEAL QUITE CHALLENGING — THE FRAMEWORK OF BELIEF REVISION DEALS WITH INCONSISTENCY IN A PRINCIPLED WAY — WE EXTEND IT TO NETWORKS OF ONTOLOGIES.

Knowledge acquired from other agents or from the environment may contradict an agent's beliefs. It may also happen that the environment changes and that the knowledge in an ontology is not up-to-date. When ontologies are connected in a network of ontologies, the addition of a new information in one ontology or the addition of a correspondence in an alignment may have long-range consequences.

One of these consequences is that the network of ontology satisfies no model: it is inconsistent. A consistent agent cannot hold inconsistent beliefs. Then it will address this problem by revising its beliefs according to the new information. In doing so, it will try to preserve as much as possible non contradicting beliefs, yet restoring consistency.

5.1 Inconsistency, isolation, repair

As we have seen (Appendix A), an ontology may be inconsistent, i.e. have no model. This means that it cannot be the representation of anything. Networks of ontologies may also be inconsistent, either because one of their ontologies is inconsistent or because they are not compatible through the alignments.

Because inconsistent knowledge has no model, any formula is a consequence of it and any answer is good for a query. Therefore, inconsistency hinders the capability to use knowledge.

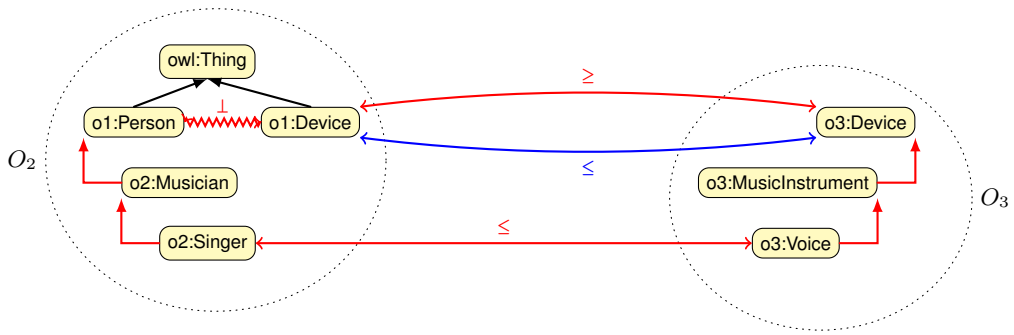


Figure 5.1. Two ontologies and one alignment which are together incoherent (o2:Singer cannot have an instance) but independently coherent.

One way of reacting to inconsistency is to try to repair the affected ontology and alignment. This may be achieved by suppressing some elements. In a network of ontologies, it may be achieved by retracting incon-

sistent ontologies or alignments. This would lead to adopt a maximal consistent subnetwork of a particular network of ontology. However, this leads to two problems: (*a*) the retraction may lead to miss many consistent knowledge, and (*b*) there may be several such sub-network.

A more precise approach would be to identify and retract faulty axioms in ontologies and faulty correspondences in alignments. However, this raises the same problems.

This may be illustrated by Figure 5.1. In this pair of aligned ontology fragments, the class o2:Singer is incoherent: it is necessarily empty. It would be inconsistent if there were an instance of o2:Singer.

However, each ontology taken in isolation is coherent. This is also true of the alignment if no constraints on the ontologies is known. There are several ways to fix this incoherence. One can modify the alignment (suppressing $\langle$o1:Device, $\geq$, o3:Device$\rangle$ or $\langle$o2:Singer, $\leq$, o3:Voice$\rangle$) or the ontologies (o1:Person$\perp$o1:Device, o2:Musician $\sqsubseteq$ o1:Person or o2:Singer $\sqsubseteq$ o2:Musician from the left-hand side ontology or o3:MusicInstrument $\sqsubseteq$ o3:Device or o3:Voice $\sqsubseteq$ o3:MusicInstrument from the right-hand side ontology).

Any of these would restore coherence. However, without further feedback, it is not easy to decide which of these solutions to select. Moreover, it may be sometimes necessary to combine the retraction of several statements. A cautious approach would consist of retracting all these statements, clearly weakening the knowledge.

In addition, depending on the retracted statements some information that is not contradictory may be lost. For instance, if one decides that the problem should be solved by suppressing the assertion o3:MusicInstrument $\sqsubseteq$ o3:Device, then it will apply to all those music instruments which are not voices, such as haegum, the information that they are music instruments, which does not raise any problem, is not entailed any more. Hence, a minimal correction of the problem, should also add more statements in order to preserve that information.

Consequently, restoring consistency is a complex task, subject to antagonist constraints. It is better addressed in a principled way as belief revision does.

5.2 Belief revision

The field of belief revision has considered the kind of problems that we face in a logical context[1]. It has proceeded by defining revision operators, constraints that they must satisfy (named postulates) and the design of such operators. We consider the basis of this field here.

5.2.1 Revision operators

Revision [Alchourrón, Gärdenfors, and Makinson 1985] considers a propositional logical language L and a closure operation[2] $Th : 2^L \to 2^L$ over sets of formulas in this language. A closed set of formulas is called a 'belief set'. The classical revision framework deals with belief sets syntactically, i.e. it considers them as closed sets of formulas, without explicit semantic assumption on the closure operation.

The use of closed sets has the advantage of providing principles which address minimality from the beginning: a closed set contains all the information to be preserved. Thus the goal of revision is simply to reduce the belief set so that it does not contain undesirable knowledge.

Revision is based on operators for modifying a belief set (K) when some action has to occur with respect to a formula (ϕ):

revision when ϕ has to be believed ($\dot{+}$),

contraction when ϕ has to be not believed ($\dot{-}$), or

update when ϕ does not hold any more ($\dot{\times}$).

These operators are constrained by postulates.

Besides 'belief revision' stricto sensu, i.e. theory revision, there exists the notion of 'base revision' [Katsuno and Mendelzon 1991b] which operates on sets of axioms not necessarily closed. These have been recently convincingly unified in [Falakh, Rudolph, and Sauerwald 2023].

[1] 'Belief revision' is used here as a generic term which will cover what we call theory revision and base revision.

[2] The original formulation uses Cn instead of Th. We use Th in order to distinguish it with the semantically defined consequence relation Cn and to emphasize that it is not a semantic operation.

5.2.2 AGM postulates for belief revision

We follow the 6 basic AGM postulates for revision [ALCHOURRÓN, GÄRDENFORS, and MAKINSON 1985] with, in parenthesis, the labels of the classification of Sven Ove Hansson [HANSSON 1996; FERMÉ and HANSSON 2018].

Here, $K + \phi$ is used to denote $Th(K \cup \{\phi\})$ (a new belief set).

Definition 5.1 (Standard AGM revision postulates). *Given a belief set $K \subseteq L$ and a formula $\phi \in L$, the belief set $K \dot{+} \phi$ resulting from revising K with ϕ must satisfy the following postulates:*

$\dot{+}$1 (closure) *$K \dot{+} \phi \supseteq Th(K \dot{+} \phi)$, i.e. the result is deductively closed;*

$\dot{+}$2 (success) *$\phi \in K \dot{+} \phi$, i.e. the new formula should be believed;*

$\dot{+}$3 (inclusion) *$K \dot{+} \phi \subseteq K + \phi$, i.e. it should not provide more knowledge than the mere addition of the formula;*

$\dot{+}$4 (vacuity) *If $\neg\phi \notin Th(K)$ then $K \dot{+} \phi \supseteq K + \phi$, i.e. if the formula is compatible with current beliefs, then simply add it;*

$\dot{+}$5 (consistency) *if $\neg\phi \notin Th(\varnothing)$, then $K \dot{+} \phi$ is consistent, i.e. revision should not bring inconsistency;*

$\dot{+}$6 (extensionality) *If $\phi \equiv \psi$, then $K \dot{+} \phi \equiv K \dot{+} \psi$, i.e. revision should be syntax-independent.*

$\dot{+}$5 mentions consistency, but this could be replaced by $\phi \wedge \neg\phi \notin K \dot{+} \phi$.

In addition to these six *basic postulates*, the AGM framework comprises two *supplementary postulates* which are dependent on the logical language used: they express the relation between revision and conjunction. They can be expressed in this way:

Definition 5.2 (Additional AGM revision postulates).

$\dot{+}$7 (superexpansion) $K \dot{+} (\phi \wedge \psi) \subseteq (K \dot{+} \phi) + \psi$;

$\dot{+}$8 (subexpansion) *if $\neg\psi \notin K \dot{+} \phi$, then* $K \dot{+} (\phi \wedge \psi) \supseteq (K \dot{+} \phi) + \psi$.

Since we are relying on ontology languages which may not have explicit conjunction, the basic postulates will mainly be considered.

This framework has been criticised and improved by other authors, and many other operators have been proposed in different contexts [GÄRDENFORS 1992]. However, the original idea has remained because its simple definition of a few independent postulates is very appealing.

5.2.3 Contraction, update and base revision

The contraction operator ($K \dot{-} \phi$, i.e. what happens when removing an assertion if one wants to ensure that it is not deducible anymore) is also considered. Contraction axioms corresponding to revision axioms can be specified:

Definition 5.3 (AGM contraction postulates).

$\dot{-}$1 (closure) *$K \dot{-} \phi \supseteq Th(K \dot{-} \phi)$, i.e. the result is deductively closed;*

$\dot{-}$2 (success) *if $\phi \notin Th(\varnothing)$, then $\phi \notin K \dot{-} \phi$, i.e. unless a tautology, the retracted formula should not be believed;*

$\dot{-}$3 (inclusion) *$K \dot{-} \phi \subseteq K$, i.e. it should not provide more knowledge than the initial theory;*

$\dot{-}$4 (vacuity) *If $\phi \notin K$ then $K \dot{-} \phi \supseteq K$, i.e. if the formula is not entailed by current beliefs, then they do not change;*

$\dot{-}$5 (recovery) *$K \subseteq (K \dot{-} \phi) + \phi$, i.e. contraction should not suppress more than can be recovered;*

$\dot{-}$6 (extensionality) *If $Th(\phi) = Th(\psi)$, then $K \dot{-} \phi = K \dot{-} \psi$, i.e. contraction should be syntax-independent.*

It has been shown that revision and contraction are inter-definable through the Lévi identity:

$$K \dot{+} \phi = (K \dot{-} \neg\phi) + \phi$$

and its converse, the Harper identity:

$$K \dot{-} \phi = Th(K \cap (K \dot{+} \neg\phi))$$

The same kind of framework has been introduced for defining an update operator ($K \dot{\times} \phi$) [KATSUNO and MENDELZON 1991a]. Revision and update can be contrasted by the following: revision operators change beliefs when knowledge about the domain has changed; update operators change beliefs when the modelled domain has changed.

Definition 5.4 (Update). update (theory -)—emph *Update can be defined by the same postulates as revision, but ($\dot{+}4$) which has to be changed for:*
$\dot{\times}$4 *If $\phi \in K$, then $K \dot{\times} \phi = K$, i.e. if the formula is already believed, then there is nothing to update;*

Update has some differences with revision. For instance, the update will not restore the consistency of an already inconsistent theory. Another difference between revision and update is that we can expect revision to not depend on the order of revision, i.e. it commutes ($K \dot{+} \phi \dot{+} \psi = K \dot{+} \psi \dot{+} \phi$), while update*update (theory -)* does not necessarily have to be order-independent.

We focus on belief revision because, in the context of open systems, the main problem is to accumulate knowledge rather than contract it. Such operators still have their utility and could certainly be developed along the methodology presented here. We do not discuss these extensions further, instead we consider applying the most standard theory to networks of ontologies.

Related principles have been proposed to account for belief base revision [NEBEL 1994] which, contrary to theory revision, does not apply to closed sets of formulas but sets of axioms. Table 5.1 provide an idea of the differences between the two approaches.

	theory revision	base revision
size	large	small
entailment	membership ($\in$)	entailment test ($\models$)
equivalence	equality ($=$)	mutual entailment ($\equiv$)
consistency	inclusion of inconsistent statement	consistency test
	$Th(\cdot) = L$	$\mathcal{M}(\cdot) = \varnothing$
natural operator	partial-meet	faithful ordered

Table 5.1. The differences between theory revision and base revision.

Base revision has the advantage of being more practical (because users and systems do not manipulate closures). However, it takes the risk to depend on syntax. Indeed, different sets of axioms may have the same closure, i.e. the same meaning, and it is preferable that the revision has the same effect on both. Since bases are not closed, it is also less convenient to compare them on the basis of inclusion: two equivalent bases may not be included in one another.

Hence, it is more convenient to define base revision semantically: using entailment instead of inclusion. This is what has been done:

Definition 5.5 (KM postulates for base revision [KATSUNO and MENDELZON 1991b]). *Given $K \subseteq L$ a set of assertions and $\phi \in L$ a formula, the set of assertions $K \bar{+} \phi$ resulting from revising K with a new belief ϕ must satisfy the postulates:*
$\bar{+}$1 (success) *$K \bar{+} \phi \models \phi$;*
$\bar{+}$2 (vacuity) *if $K \cup \{\phi\}$ is consistent, then $K \bar{+} \phi \equiv K \cup \{\phi\}$;*
$\bar{+}$3 (consistency) *if ϕ is consistent, then $K \bar{+} \phi$ is consistent;*
$\bar{+}$4 (extensionality) *if $K \equiv K'$ and $\phi \equiv \psi$ then $K \bar{+} \phi \equiv K' \bar{+} \psi$;*
$\bar{+}$5 (superexpansion) *$(K \bar{+} \phi) \cup \{\psi\} \models K \bar{+} (\phi \wedge \psi)$;*
$\bar{+}$6 (subexpansion) *if $(K \bar{+} \phi) \cup \{\psi\}$ is consistent, then $K \bar{+} (\phi \wedge \psi) \models (K \bar{+} \phi) \cup \{\psi\}$.*

Here, the use of $\cup$ replaces that of $+$, because closure is not required (the authors did not use $\cup$ for another technical reason). Closure is not needed any more because belief bases are not closed and inclusion, which aimed at confining the revision within the closure, is avoided: indeed, it may be necessary to add axioms which

were previously entailed but were not included in the base. On the contrary, extensionality did not had to be asserted for theories because two theories are equivalent if they are equal (their closure is the same), though this does not hold for bases.

These postulates characterise semantically operators equivalent to theory revision.

Proposition 5.6 ([KATSUNO and MENDELZON 1991b]). *$\bar{+}$ is a base revision satisfying $\bar{+}1$–$\bar{+}4$ iff the theory revision operator defined as $Cn(K\bar{+}\phi)$ satisfies $\dot{+}1$–$\dot{+}6$. $\bar{+}5$–$\bar{+}6$ are then equivalent to $\dot{+}7$–$\dot{+}8$.*

The authors also show that for propositional logics with a finite signature, base revision operators are exactly those which return the minimal models of ϕ for a K-induced faithful preorder over models.

Hereafter, we consider theory revision instead of base revision. However, while the reasoning applies to closed sets, examples are always provided on bases (because closed sets would be too large).

5.2.4 Partial-meet revision

Intuitively, for applying revision, it is necessary to suppress from the closure assertions that (*a*) conflict with the new assertion, (*b*) cannot be deduced any more, (*c*) so that adding the new assertion to the result is consistent. Partial-meet revision [ALCHOURRÓN, GÄRDENFORS, and MAKINSON 1985] is a very general form of revision operator. It is based on the maximal subsets of K which remain consistent when adding ϕ (denoted by $K\top\phi$):

$$K\top\phi = \{K' \subseteq K \mid \neg\phi \notin Th(K') \text{ and } \forall K''; K' \subset K'' \subseteq K, \neg\phi \in Th(K'')\}$$

Different operators may be obtained by selecting γ some of these subsets and intersecting them. Partial-meet revision is defined by:

$$K\dot{+}\phi = \bigcap \gamma(K\top\phi) + \phi$$

such that if $K\top\phi \neq \varnothing$, then $\gamma(K\top\phi) \neq \varnothing$.

When γ selects all sets in $K\top\phi$, then the operator is called full-meet revision and when it selects only one such set, it is the maxichoice revision.

Partial-meet revision satisfy all basic postulates.

Proposition 5.7 (Satisfaction [ALCHOURRÓN, GÄRDENFORS, and MAKINSON 1985]). *Partial-meet revision operators satisfy postulates $\dot{+}1$–$\dot{+}6$.*

Partial-meet revision operators are among the most natural such operators.

5.3 Adaptation to ontologies and alignments

Revision operators have to be adapted to their use in networks of ontologies. Although it seems that they can be used right away with ontologies, some issues have to be considered before this can be possible (§5.3.1). Dealing with them allows for designing ontology revision operators (§5.3.2). This enable revising alignments and introduce corresponding partial-meet revision operators (§5.3.3).

5.3.1 Problems for the application to description logics

Several problems may be raised to adopt the revision framework for ontologies and alignments [FLOURIS, PLEXOUSAKIS, and ANTONIOU 2006]:

- they do not have a negation of a formula, which renders the Levi and Harper identities non applicable;
- they do not have a negation of a formula, which makes the classical formulation of $\dot{+}4$, $\dot{+}5$ and $\dot{+}8$ non expressible;
- they do not have the conjunction of formulas, which prevents to express $\dot{+}7$ and $\dot{+}8$ (and $\bar{+}5$ and $\bar{+}6$ for base revision).

In fact, the revision postulates mix a set notation ($\cup$ often hidden in $+$) and a logic notation ($\wedge$). This had been simplified in [KATSUNO and MENDELZON 1991b] through always using $\wedge$: a base would be expressed as a conjunction of the formulas in the base. Hence, the base becomes a formula and it is possible to revise by a set of formulas by making it a conjunction. Semantically, the meaning is the same.

However, for languages without formula conjunction, like ontology and alignment languages, this may be difficult. It is possible to proceed in the opposite way and to express conjunction with sets of formulas. In principle, the same level of generality can be obtained by allowing the revision by a set of formula instead of one single formula [FALAKH, RUDOLPH, and SAUERWALD 2023]. Here, we will systematically use sets instead of conjunction, which aligns well with decomposing everything in sets of triples or correspondences, but revise by a single formula at a time.

AGM uses a closure operation which is a purely syntactic operator (Th), but it is possible to consider it semantically (Cn) [DALAL 1988]. If this closure is tied to semantics by completeness ($Th \equiv Cn$), then postulates can be reformulated in model-theoretic terms. This addresses some of the issues raised by the absence of negation: the use of negation in $\dot{+}4$, $\dot{+}5$ and $\dot{+}8$ are expression of consistency which are better expressed in these terms. This does not, however, solve the problem with the Levi and Harper identities.

Moreover, in this course, we are attached to define the semantics of presented concepts. Hence, we take a semantic point of view and deal with semantic consequence when expressing postulates for ontologies and alignments.

5.3.2 Operators for description logic ontologies

To introduce semantic revision postulates, the following changes are made:
- the syntactic closure function (Th) is replaced by the semantic one (Cn^ω);
- consistency/inconsistency is expressed as the existence/absence of models;
- consistency is used instead of $\neg$.

This results in the following definition as reformulated by Bernhard Nebel [NEBEL 1994].

Definition 5.8 (Semantic revision postulates for ontologies). *Given a closed ontology O and an ontology axiom δ, the ontology $O\dot{+}\delta$ resulting from revising O with δ must satisfy the following postulates:*

$\dot{+}$1 (closure) *$O\dot{+}\delta \supseteq Cn^\omega(O\dot{+}\delta)$;*

$\dot{+}$2 (success) *$\delta \in O\dot{+}\delta$;*

$\dot{+}$3 (inclusion) *$O\dot{+}\delta \subseteq O + \delta$;*

$\dot{+}$4 (vacuity) *If $O + \delta$ is consistent, then $O\dot{+}\delta \supseteq O + \delta$;*

$\dot{+}$5 (consistency) *If $O\dot{+}\delta$ is inconsistent, then δ is inconsistent;*

$\dot{+}$6 (extensionality) *If $\delta \equiv \delta'$, then $O\dot{+}\delta \equiv O\dot{+}\delta'$.*

Similarly, for partial-meet revision, $\top$ may be rewritten semantically without negation as:

$$O\top\delta = \{O' \subseteq O \mid \mathcal{M}(O' + \delta) \neq \varnothing \text{ and } \forall O''; O' \subset O'' \subseteq O, \mathcal{M}(O'' + \delta) = \varnothing\}$$

5.3.3 Alignment revision

As agents interact, they may have to expand the alignments between their ontologies. Like there are revision operators for ontologies, it is possible to consider an alignment revision operator whose goal is to add a new correspondence to an alignment.

We have not seen that alignments may become inconsistent independently from the ontologies they relate. For an alignment to be inconsistent it is necessary that there is no model for this alignment. This may be the case in the situation described by Figure 5.2: the alignment between O and O' asserts that the same ($=$) individual, denoted by different IRIs (o and o') belongs ($\in$) to two disjoint ($\bot$) classes (c and c'). Independently from any constraint on the ontologies, this cannot have a model. More realistically, inconsistency may be raised when using more elaborate alignment languages.

Here are the corresponding postulates (as before, $A + \mu$ denotes $Cn^\alpha(A \cup \{\mu\})$):

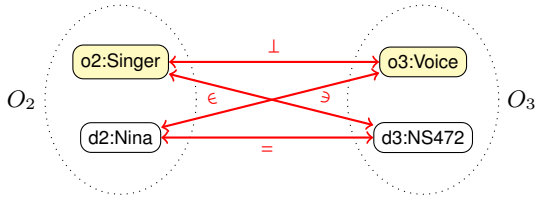


Figure 5.2. Locally inconsistent alignment: an inconsistent alignment that can only be made consistent by modifying it.

Definition 5.9 (Semantic revision postulates for alignments). *Given a closed alignment A and a correspondence μ, the alignment $A\dot{\oplus}\mu$ resulting from revising A with μ must satisfy the following postulates:*
$\dot{\oplus}$1 (closure) $A\dot{\oplus}\mu \supseteq Cn^{\alpha}(A\dot{\oplus}\mu)$
$\dot{\oplus}$2 (success) $\mu \in A\dot{\oplus}\mu$, *i.e.* $A\dot{\oplus}\mu \models \mu$
$\dot{\oplus}$3 (inclusion) $A\dot{\oplus}\mu \subseteq A + \mu$
$\dot{\oplus}$4 (vacuity) *If* $A + \mu$ *consistent, then* $A\dot{\oplus}\mu \supseteq A + \mu$;
$\dot{\oplus}$5 (consistency) *If* $A\dot{\oplus}\mu$ *inconsistent, then* μ *is inconsistent*
$\dot{\oplus}$6 (extensionality) *If* $\mu \equiv \nu$ *then* $A\dot{\oplus}\mu \equiv A\dot{\oplus}\nu$

Such a revision operator is local in two senses: it is constrained only by the alignment and the two ontologies, and it only modifies the alignment.

All ingredients for defining partial-meet revision on alignments are readily available: (*a*) alignments can be considered as closed by the Cn^{α} function; (*b*) models for alignments have been defined (§3.1.2); (*c*) inclusion for defining $\top$ is simply set inclusion.

Hence, a partial-meet revision operator on alignments can be defined as:

$$A\dot{\oplus}\mu = \bigcap \gamma(A\top\mu) + \mu$$

such that

$$A\top\mu = \{A' \subseteq A \mid \mathcal{M}(A' + \mu) \neq \varnothing \text{ and } \forall A''; A' \subset A'' \subseteq A, \mathcal{M}(A'' + \mu) = \varnothing\}$$

Proposition 5.10 ([Euzenat 2015]). *Partial-meet revision operators on alignments satisfy the six basic postulates for alignment revision $\dot{\oplus}1$–$\dot{\oplus}6$.*

The question that we consider is: can AGM revision postulates be applied to networks of ontologies?

5.4 Revision operators for networks of ontologies

We now can deal with revision operators for networks of ontologies. We first introduce specific operators for dealing with networks of ontologies (§5.4.1): they depend on the type of formula by which the network is revised (ontology axiom or correspondence). We then provide postulates for revising networks of ontologies depending on the type of formula used for revising: an ontology formula or a correspondence (§5.4.2). We then consider the definition of global revision operators based on the regular ontology revision operators and this alignment revision operator and provide their properties (§5.4.3). Although such an operator can be defined, it would be too local (disallowing ontology revision by an alignment change or vice versa). Hence we introduce a partial-meet revision operator for networks of ontologies that works globally (§5.4.4).

5.4.1 Network of ontology revision operators

Revision may be needed at two different occasions: (*a*) When adding a formula δ to an ontology O, or (*b*) When adding a correspondence μ to an alignment A. Revision of a network by adding ontologies and alignments can also be considered. However, adding new ontologies does not bring new problems if they are consistent. Since they are not yet connected to other ontologies, they will be considered as connected through empty alignments.

We do not consider the operations of adding new ontologies or alignments, because it is always possible to introduce empty ontologies and alignments, which do not create inconsistency and to add axioms and correspondences with the corresponding operators. Then adding alignments will be like revising by modifying these initially empty alignments.

Hence, we only have to consider two operators:

Definition 5.11 (Revision operators in a network of ontologies). *Given a closed network of ontologies $\langle \Omega, \Lambda \rangle$, two types of revision operators are distinguished:*
– $\langle \Omega, \Lambda \rangle \dot{\boxplus} \delta / O$*: revising the ontology O by adding the formula δ;*
– $\langle \Omega, \Lambda \rangle \dot{\boxplus} \mu / A$*: revising the alignment A by adding the correspondence μ;*
with $O \in \Omega$ and $A \in \Lambda$.

Inconsistency may manifest itself in three ways (§3.3):
ω-local Because an ontology has no (local) model anymore (only with the first operation);
α-local Because an alignment is inconsistent in itself (§5.3.3);
global Because there is no model combination that satisfies all the alignments.
An operation can only trigger one kind of local inconsistency or global inconsistency.

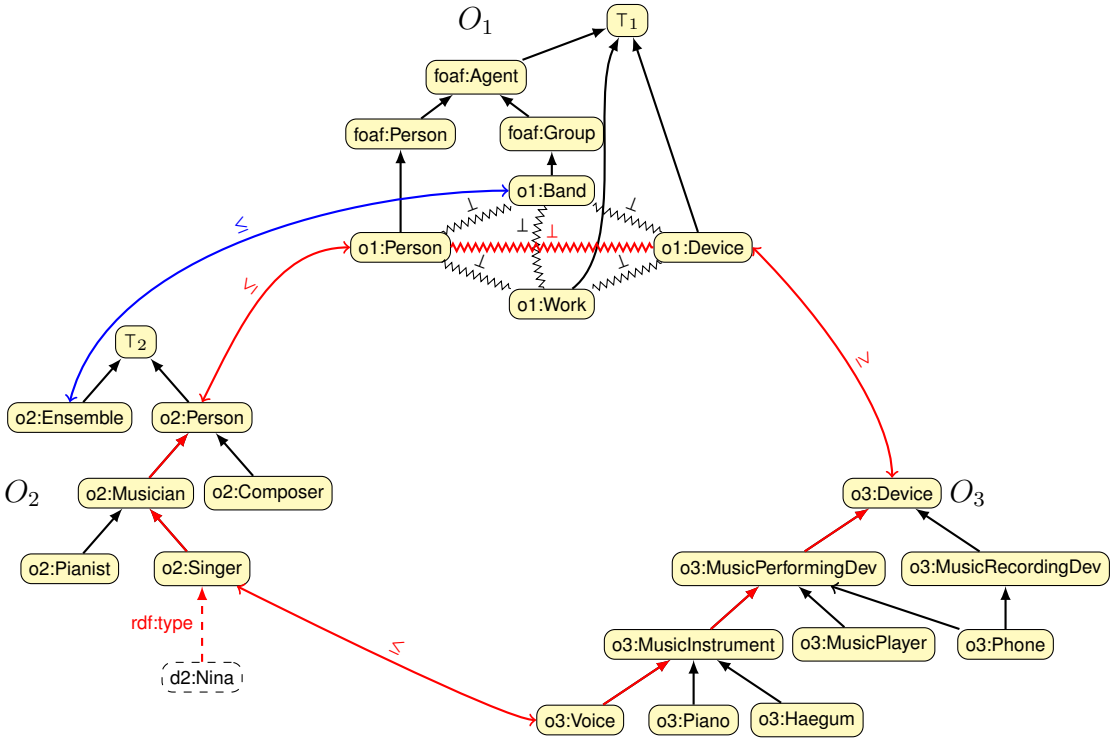


Figure 5.3. An inconsistent network of ontologies N_2 made of three ontologies (O_1, $O_2 \cup \{\mathsf{d2{:}Nina} \sqsubseteq \mathsf{o2{:}Singer}\}$, and O_3) and three alignments ($A_{1,2}$, $A_{1,3}$, and $A_{2,3}$). In red, the statements involved in the inconsistency.

In the following examples, we start with the inconsistent network of ontologies N_2 of Figure 5.3. This may be the result of different actions modifying a previous network:

0. $\langle \{O_1 \cup \{\mathsf{d2{:}Nina} \sqsubseteq \mathsf{o1{:}Person}\}, O_2, O_3\}, \{A_{1,2}, A_{1,3}, A_{2,3}\} \rangle \dot{\boxplus} (\mathsf{d2{:}Nina} \sqsubseteq \mathsf{o1{:}Band}) / O_1$
This is an ontology revision that raises an ω-local inconsistency.
1. $\langle \{O_1, O_2, O_3\}, \{A_{1,2}, A_{1,3}, A_{2,3}\} \rangle \dot{\boxplus} (\mathsf{d2{:}Nina} \sqsubseteq \mathsf{o2{:}Singer}) / O_2$
This is an ontology revision that raises a global inconsistency (illustrated in Figure 5.3).
2. $\langle \{O_1 \setminus \{\mathsf{o1{:}Person} \sqsubseteq \mathsf{o1{:}Device}\}, O_2 \cup \{\mathsf{d2{:}Nina} \sqsubseteq \mathsf{o1{:}Singer}\}, O_3\}, \{A_{1,2}, A_{1,3}, A_{2,3}\} \rangle \dot{\boxplus} (\mathsf{o1{:}Person} \sqsubseteq \mathsf{o1{:}Device}) / O_1$
This raises a global inconsistency but no local inconsistency as O_1 alone is consistent.

– *Pure α-local inconsistency is more difficult to encounter due to the little expressiveness of alignments. However, Figure 5.2 illustrates this situation: the ontologies contain no assertions (beside that there are classes and individuals) and the alignment imposes constraints that can be satisfied by no pair of models.*

3. $\langle\{O_1, O_2, O_3\}, \{A_{1,2}, A_{1,3}\}\rangle \dot{\boxplus}(\text{o2:Singer} \leq \text{o3:Voice})/A_{2,3}$
This is an alignment change that raises a global inconsistency.

Finally, revision can be achieved by two means:

– By retracting and adding axioms ($\langle \Omega \pm \delta'/O', \Lambda\rangle$) for global and ω-local inconsistency;
– By retracting and adding correspondences ($\langle \Omega, \Lambda \pm \mu'/A'\rangle$) for global and α-local inconsistency.

We consider the addition of axioms and correspondences within revision, because in base revision, it may be necessary to complete the base so that it looses as little information as possible. However, in theory revision, addition is useless because theories are closed.

In the examples above, from the inconsistent network of ontologies N_2 of Figure 5.3, consistency may be recovered in various ways:

0. $\langle\{O_1 \cup \{\text{d2:Nina} \sqsubseteq \text{o1:Person}\}, O_2, O_3\}, \{A_{1,2}, A_{1,3}, A_{2,3}\}\rangle \dot{\boxplus}(\text{d2:Nina} \sqsubseteq \text{o1:Band})/O_1$
The ω-local inconsistency may be solved by removing o1:Person$\sqcap$o1:Band *or* d2:Nina $\sqsubseteq$ o1:Person *from O_1, but cannot be solved by removing anything from the other ontologies or alignments.*

1. $\langle\{O_1, O_2, O_3\}, \{A_{1,2}, A_{1,3}, A_{2,3}\}\rangle \dot{\boxplus}(\text{d2:Nina} \sqsubseteq \text{o2:Singer})/O_2$
The global inconsistency may be solved locally by removing any statement of O_2 marked in red in Figure 5.3: o2:Singer $\sqsubseteq$ o2:Musician *(and its consequences) or* o2:Musician $\sqsubseteq$ o2:Person *(and its consequences), but it can also be solved more globally, for instance, by removing* o1:Person $\geq$ o2:Person *from $A_{1,2}$ (and its consequences),* o3:Singer $\leq$ o3:Voice *from $A_{2,3}$ (and its consequences) or even* o1:Device $\geq$ o3:Device *from $A_{1,3}$ (and its consequences), etc. In each case, the discarded consequences span beyond the local changes, in general they would comprise* d2:Nina $\sqsubseteq$ o1:Person *or* d2:Nina $\sqsubseteq$ o1:Device. *If there were many* o2:Singers, *it would be wiser to modify an ontology statement than each singer situation individually.*

2. $\langle\{O_1 \setminus \{\text{o1:Person}\sqcap\text{o1:Device}\}, O_2 \cup \{\text{d2:Nina} \sqsubseteq \text{o1:Singer}\}, O_3\}, \{A_{1,2}, A_{1,3}, A_{2,3}\}\rangle \dot{\boxplus}(\text{o1:Person}\sqcap\text{o1:Device})/O_1$
The global (and non local) inconsistency but no local inconsistency cannot be solved by modifying O_1, but by suppressing any of the other statements in red (and their consequences) also leading to suppress either d2:Nina $\sqsubseteq$ o1:Person *or* d2:Nina $\sqsubseteq$ o1:Device.

– *A pure α-local inconsistency such as the one in Figure 5.2 may be recovered by removing any of the four initial correspondences and only in this way.*

3. $\langle\{O_1, O_2, O_3\}, \{A_{1,2}, A_{1,3}\}\rangle \dot{\boxplus}(\text{o2:Singer} \leq \text{o3:Voice})/A_{2,3}$
The global inconsistency cannot be solved locally, as $A_{2,3}$ was previously empty and thus consistent. This can only be solved by suppressing any of the other statements in red (and their consequences).

Cases 2 and 3 have been chosen so that the global inconsistency cannot be solved locally, but it may happen that global inconsistencies can be solved locally.

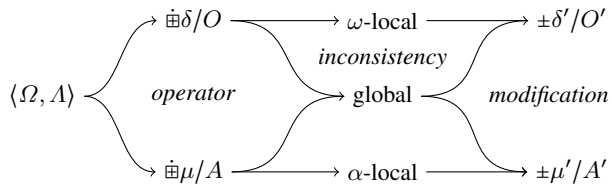


Figure 5.4. Phenomena occurring when revising networks of ontologies.

A summary of the situation is presented in Figure 5.4: the two possible operators (ontology or alignment revision) may entail three different types of inconsistency (two local and one global inconsistency). These three types of inconsistency can be resolved by two types of modifications (in ontologies and/or alignments). Local inconsistencies may only be raised by revision of the concerned ontology or alignment and only be solved by applying changes to this ontology or alignment. Global inconsistency may be raised and solved by both operations, independently.

5.4.2 Revision postulates for networks of ontologies

Revision operators, both for alignments and ontologies, must be global, i.e. such that the closure operator in the postulates is defined with respect to the whole network of ontologies. Networks of ontologies have already been defined as a logic with a consequence closure (Cn).

However, we need a relation for comparing networks such as inclusion or subsumption. In the general case, this may reveal difficult [EUZENAT 2014b]. However, when dealing with revision of networks of ontologies we can consider networks with which there exist an isomorphism between the initial network and those it is compared and combined with. To enforce compatibility between these isomorphisms, we will consider networks of ontologies with the same number of ontologies indexed by a finite set Ξ. These networks will also be considered normalised, i.e. for each pair of ontology O and O' there exists exactly one alignment between them called $\lambda(O, O')$, eventually empty. It is always possible to normalise networks of ontologies and the normal form has the same set of models. In addition, for theory revision, these networks are closed. The set of such networks will be called $\mathcal{N}_\Xi$.

It is now possible to introduce subsumption between these networks. It is called syntactic subsumption, because it assumes that the compared ontologies use the same vocabulary, hence use simple inclusion.

Definition 5.12 (Syntactic subsumption). *$\langle \Omega, \Lambda\rangle \in \mathcal{N}_\Xi$ is (syntactically) subsumed by $\langle \Omega', \Lambda'\rangle \in \mathcal{N}_\Xi$ (noted $\langle \Omega, \Lambda\rangle \sqsubseteq \langle \Omega', \Lambda'\rangle$) if*
- $\forall i \in \Xi$, $O_i \subseteq O'_i$, *and*
- $\forall i, j \in \Xi$, $\lambda(O_i, O_j) \subseteq \lambda'(O'_i, O'_j)$

Similarly as before, two addition operations (note the lack of the dot on $\boxplus$) introduce an axiom in an ontology or a correspondence in an alignment:

$$\langle \Omega, \Lambda\rangle \boxplus \delta/O = Cn(\langle \Omega \setminus \{O\} \cup \{O + \delta\}, \Lambda\rangle)$$

and

$$\langle \Omega, \Lambda\rangle \boxplus \mu/A = Cn(\langle \Omega, \Lambda \setminus \{A\} \cup \{A + \mu\}\rangle)$$

Naturally, $\langle \Omega, \Lambda\rangle$, $\langle \Omega, \Lambda\rangle \boxplus \delta/O$ and $\langle \Omega, \Lambda\rangle \boxplus \mu/A$ belong to a same $\mathcal{N}_\Xi$ with the same index for ontologies (and O having the remaining index). Thus, $\langle \Omega, \Lambda\rangle \sqsubseteq \langle \Omega, \Lambda\rangle \boxplus \delta/O$ and $\langle \Omega, \Lambda\rangle \sqsubseteq \langle \Omega, \Lambda\rangle \boxplus \mu/A$

An operator $\dot{\boxplus}$, such that $\langle \Omega, \Lambda\rangle \dot{\boxplus} \mu/A$ is the result of the revision of $\langle \Omega, \Lambda\rangle$ by a new correspondence μ in the alignment $A \in \Lambda$, can be defined in the AGM postulate style:

Definition 5.13 (Semantic alignment revision postulates for networks of ontologies). *Given a closed network of ontologies $\langle \Omega, \Lambda\rangle$, a closed alignment $A \in \Lambda$ and a correspondence μ, the network of ontologies $\langle \Omega, \Lambda\rangle \dot{\boxplus} \mu/A$ resulting from the revision of A by μ must satisfy the following postulates:*

$\dot{\boxplus}$1 (closure) $\langle \Omega, \Lambda\rangle \dot{\boxplus} \mu/A \supseteq Cn(\langle \Omega, \Lambda\rangle \dot{\boxplus} \mu/A)$;

$\dot{\boxplus}$2 (success) $\langle \Omega, \Lambda\rangle \dot{\boxplus} \mu/A \models \mu$

$\dot{\boxplus}$3 (inclusion) $\langle \Omega, \Lambda\rangle \dot{\boxplus} \mu/A \sqsubseteq \langle \Omega, \Lambda\rangle \boxplus \mu/A$;

$\dot{\boxplus}$4 (vacuity) *If $\langle \Omega, \Lambda\rangle \boxplus \mu/A$ consistent, then* $\langle \Omega, \Lambda\rangle \dot{\boxplus} \mu/A \supseteq \langle \Omega, \Lambda\rangle \boxplus \mu/A$;

$\dot{\boxplus}$5 (consistency) *If $\langle \Omega, \Lambda\rangle \dot{\boxplus} \mu/A$ inconsistent, then μ is inconsistent;*

$\dot{\boxplus}$6 (extensionality) *If $\mu \equiv \nu$, then* $\langle \Omega, \Lambda\rangle \dot{\boxplus} \mu/A \equiv \langle \Omega, \Lambda\rangle \dot{\boxplus} \nu/A$;

The same can be defined for ontology revision.

Definition 5.14 (Semantic ontology revision postulates for networks of ontologies). *Given a closed network of ontologies $\langle \Omega, \Lambda\rangle$, a closed ontology $O \in \Omega$ and a formula δ, the network of ontologies $\langle \Omega, \Lambda\rangle \dot{\boxplus} \delta/O$ resulting from the revision of O by δ must satisfy the following postulates:*

$\dot{\boxplus}$1 (closure) $\langle \Omega, \Lambda\rangle \dot{\boxplus} \delta/O \supseteq Cn(\langle \Omega, \Lambda\rangle \dot{\boxplus} \delta/O)$

$\dot{\boxplus}$2 (success) $\langle \Omega, \Lambda\rangle \dot{\boxplus} \delta/O \models \delta/O$

$\dot{\boxplus}$3 (inclusion) $\langle \Omega, \Lambda\rangle \dot{\boxplus} \delta/O \sqsubseteq \langle \Omega, \Lambda\rangle \boxplus \delta/O$

$\dot{\boxplus}$4 (vacuity) *If $\langle \Omega, \Lambda\rangle \boxplus \delta/O$ consistent, then* $\langle \Omega, \Lambda\rangle \dot{\boxplus} \delta/O \supseteq \langle \Omega, \Lambda\rangle \boxplus \delta/O$;

$\dot{\boxplus}$***5 (consistency)*** *If* $\langle \Omega, \Lambda \rangle \dot{\boxplus} \delta / O$ *inconsistent, then* δ *is inconsistent*

$\dot{\boxplus}$***6 (extensionality)*** *If* $\delta \equiv \epsilon$*, then* $\langle \Omega, \Lambda \rangle \dot{\boxplus} \delta / O \equiv \langle \Omega, \Lambda \rangle \dot{\boxplus} \epsilon / O$

In both cases, the principles are the same as for the local postulates:

$\dot{\boxplus}$***1 (closure)*** The revised network is closed (this is theory revision);

$\dot{\boxplus}$***2 (success)*** The new axiom or correspondence is consequence of the network after revision;

$\dot{\boxplus}$***3 (inclusion)*** The consequences of the revision cannot extend beyond those of the addition;

$\dot{\boxplus}$***4 (vacuity)*** The revised network is equivalent to the closure of the one resulting from the addition, unless the latter is inconsistent;

$\dot{\boxplus}$***5 (consistency)*** The only reason why the result of revision may be inconsistent, is because the new axiom or correspondence is itself inconsistent;

$\dot{\boxplus}$***6 (extensionality)*** Revision is syntax-independent.

5.4.3 Local revision is not sufficient

A tempting approach to define revision operators for networks of ontologies would be to start from local revision operators and apply them where (in the alignment or ontology in which) revision occurs. This would have the benefit of defining revision operators for networks of ontologies only by composing local revision operators. Indeed, one would like that:

$$\langle \Omega, \Lambda \rangle \dot{\boxplus} \delta / O = Cn(\langle \Omega \setminus \{O\} \cup \{O \dot{+} \delta\}, \Lambda \rangle)$$

and, for alignment revision, that:

$$\langle \Omega, \Lambda \rangle \dot{\boxplus} \mu / A = Cn(\langle \Omega, \Lambda \setminus \{A\} \cup \{A \dot{\oplus} \mu\} \rangle)$$

Network revision operators are generalisations of their local counterparts, in the sense that, if the network of ontologies is reduced to one ontology or a pair of aligned ontologies, then the simple use of valid revision operators on ontologies or alignments are valid revision operators on the network.

Proposition 5.15 ([Euzenat 2015]). *If* $\dot{+}$ *satisfies* $\dot{+}$*1–6, then the operation* $\dot{\boxplus}$ *defined as*

$$\langle \{O\}, \varnothing \rangle \dot{\boxplus} \delta / O = Cn(\langle \{O \dot{+} \delta\}, \varnothing \rangle)$$

satisfies $\dot{\boxplus}$*1–6.*

The same applies for alignments:

Proposition 5.16 ([Euzenat 2015]). *If* $\dot{\oplus}$ *satisfies* $\dot{\oplus}$*1–6, then the operation* $\dot{\boxplus}$ *defined as*

$$\langle \{O, O'\}, \{A\} \rangle \dot{\boxplus} \mu / A = Cn(\langle \{O, O'\}, \{A \dot{\oplus} \mu\} \rangle)$$

such that A *is an alignment between* O *and* O'*, satisfies* $\dot{\boxplus}$*1–6.*

These properties are interesting in their own right but do not provide any concrete global revision operation. There are two problems related to the inconsistency and modification steps of Figure 5.4.

The first problem when considering networks of ontologies is that the notion of consistency to take into account is now related to the network of ontologies and not anymore to the locally affected ontology or alignment. It may happen that the added formula or correspondence does lead to global inconsistency but not local inconsistency (as in Cases 2 and 3 above). What happens, is that $\langle \Omega, \Lambda \rangle \dot{\boxplus} \delta / O$ is inconsistent while $O \dot{+} \delta$ is consistent. The inconsistency could be brought by the incompatibility of the remaining models of $O \dot{+} \delta$ with the rest of the network. The same may apply with alignment revision (Case 3): it may be the case that $\langle \Omega, \Lambda \rangle$ is consistent, that $A \dot{\oplus} \mu$ is consistent but that $\langle \Omega, \Lambda \rangle \dot{\boxplus} \mu / A = Cn(\langle \Omega, \Lambda \setminus \{A\} \cup \{A \dot{\oplus} \mu\} \rangle)$ is inconsistent.

This problem is relatively easy to solve in the case of partial-meet revision by replacing $\mathcal{M}(K' + \delta)$ and $\mathcal{M}(A' + \mu)$ by $\mathcal{M}(\langle \Omega, \Lambda \rangle + \delta / K')$ and $\mathcal{M}(\langle \Omega, \Lambda \rangle + \mu / A')$ in the definitions of $K \top \delta$ (§5.3.2) and $A \top \mu$ (§5.3.3) respectively.

The second problem is related to the way to apply revision: in these cases, applying local revision operators would not change the network as the local structures are consistent, and thus would result in an inconsistent network, thus thus violating $\dot{\boxplus}5$. This shows that it is not safe to revise a network of ontologies with one formula or one correspondence by only applying a local revision operator.

However, this does not prevent anyone from taking advantage of local revision operators. Indeed, if applying such a local revision operator restores consistency, then it will be a valid revision operator and even a minimal one with regard to the locality of the changes that have been made. But there may be cases where more is needed for restoring consistency.

5.4.4 Partial-meet revision operators for networks of ontologies

Partial-meet revision is the intersection (meet) of selected networks resulting from the addition of the revising statement to maximal subnetworks consistent with it.

It is possible to define maximal consistent subnetworks with respect to syntactic subsumption. There can be two maximal consistent subnetwork operators depending on what is to be revised (ontology or alignment):

Definition 5.17 (Maximal consistent subnetworks of ontologies). *Given a network of ontologies $\langle \Omega, \Lambda\rangle$, an ontology $O \in \Omega$ (resp. an alignment $A \in \Lambda$) and an assertion δ (resp. a correspondence μ), the set of maximal consistent subnetworks of $\langle \Omega, \Lambda\rangle$ with respect to δ in O (resp. μ in A) is defined by:*

$$\begin{aligned}\langle \Omega, \Lambda\rangle \Subset \delta/O = \{&\langle \Omega', \Lambda'\rangle \sqsubseteq \langle \Omega, \Lambda\rangle; \mathcal{M}(\langle \Omega', \Lambda'\rangle \boxplus \delta/O) \neq \varnothing, \text{ and}\\ &\forall \langle \Omega'', \Lambda''\rangle; \langle \Omega', \Lambda'\rangle \sqsubset \langle \Omega'', \Lambda''\rangle \sqsubseteq \langle \Omega, \Lambda\rangle, \mathcal{M}(\langle \Omega'', \Lambda''\rangle \boxplus \delta/O) = \varnothing\}\\ \langle \Omega, \Lambda\rangle \Subset \mu/A = \{&\langle \Omega', \Lambda'\rangle \sqsubseteq \langle \Omega, \Lambda\rangle; \mathcal{M}(\langle \Omega', \Lambda'\rangle \boxplus \mu/A) \neq \varnothing, \text{ and}\\ &\forall \langle \Omega'', \Lambda''\rangle; \langle \Omega', \Lambda'\rangle \sqsubset \langle \Omega'', \Lambda''\rangle \sqsubseteq \langle \Omega, \Lambda\rangle, \mathcal{M}(\langle \Omega'', \Lambda''\rangle \boxplus \mu/A) = \varnothing\}\end{aligned}$$

These operators return sets of networks of ontologies in $\mathcal{N}_\Xi$ which are subsumed by $\langle \Omega, \Lambda\rangle$ and which remain consistent when the corresponding element is added.

Consider the inconsistent network of Figure 5.3 (p.60). The axioms involved in the inconsistency are those 10 edges marked in red. The network can be the result of adding any of these statements to a network that does not contain it (p.60).

Table 5.2 shows, when suppressing each of these statements, some of the consequences that have to be suppressed with it for being a maximal consistent subnetwork of $\langle\{O_1, O_2 \cup \{G\}, O_3\}, \{A_{1,2}, A_{1,3}, A_{2,3}\}\rangle$. Only a few of these consequences are considered, there are many others. The shape of the subnetworks is very dependent of the initial network. In some cases, it may be necessary to suppress several axioms to obtain consistent subnetworks or that some involved axioms are entailed by other axioms.

Each of these subnetworks may be used in partial-meet revisions, except the one which is generated by suppressing the axiom by which the revision arises, i.e. if the inconsistency is raised through revising the network without statement o1:Person⊥o1:Device *by this very statement, then S_1 is not a maximal subnetwork consistent with* o1:Person⊥o1:Device.

The intersection of all these networks but one, will be very small.

Figure 5.5 displays an extreme example of a network. This network is inconsistent in the interpretation of Definition 3.4, 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 only be solved by suppressing one of the other *correspondences.*

For generalising partial-meet revision, we need a meet operation between networks of ontologies. Because we are dealing with very specific networks of ontologies in $\mathcal{N}_\Xi$ resulting from the reduction of $\langle \Omega, \Lambda\rangle$, we can restrict ourselves on a specific definition for meet (corresponding to the *normal fibred meet* in [Euzenat 2014b]):

$$\langle \Omega', \Lambda'\rangle \dot{\sqcap} \langle \Omega'', \Lambda''\rangle = \langle \{O'_i \cap O''_i\}_{i\in\Xi}, \{A'_{i,j} \cap A''_{i,j}\}_{i,j\in\Xi}\rangle$$

It can be checked that $\langle \Omega', \Lambda'\rangle \sqsubseteq \langle \Omega'', \Lambda''\rangle$ iff $\langle \Omega', \Lambda'\rangle = \langle \Omega', \Lambda'\rangle \dot{\sqcap} \langle \Omega'', \Lambda''\rangle$. The meet of consistent subnetworks is always consistent since it is subsumed by consistent networks (downward consistency preservation). Moreover, if networks are closed, then their meet is closed as well (because it meets closed networks). Finally, the meet of subnetworks of $\langle \Omega, \Lambda\rangle$ is still subsumed by $\langle \Omega, \Lambda\rangle$.

It is now possible to define partial-meet revision operators for networks of ontologies:

subnetwork	axiom	O_2	O_3	$A_{1,2}$			$A_{1,3}$				$A_{2,3}$	
		d2:Nina ∈ o2:Person	o2:Singer ⊑ o2:Person	o3:Voice ⊑ o3:Device	o2:Singer⊥o1:Device	o2:Singer ⊑ o1:Device	d2:Nina ∈ o1:Person	d2:Nina ∈ o1:Device	o1:Device ≥ o3:Voice	o1:Person⊥o3:Voice	o2:Singer⊥o3:Voice	d2:Nina ∈ o3:Voice
S_1	o1:Person⊥o1:Device			×					×	×		
S_2	d2:Nina ∈ o2:Singer	×				×	×					×
S_3	o2:Singer ⊑ o2:Musician	×	×	×		×					×	
S_4	o2:Musician ⊑ o2:Person	×	×	×		×					×	
S_5	o3:MusicPerformingDev ⊑ o3:Device			×		×		×	×	×	×	
S_6	o3:MusicInstrument ⊑ o3:MusicPerformingDev			×		×		×	×	×	×	
S_7	o3:Voice ⊑ o3:MusicInstrument			×		×		×	×	×	×	
S_8	o1:Person ≥ o2:Person			×		×				×	×	
S_9	o1:Device ≥ o3:Device					×		×	×	×	×	
S_{10}	o2:Singer ≤ o3:Voice					×		×				×

Table 5.2. Maximal consistent subnetworks of $\langle \Omega, \Lambda \rangle$ (obtained by removing from the closed network, an axiom and its proper consequences).

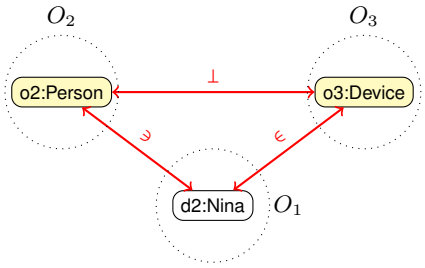


Figure 5.5. Globally inconsistent network of ontologies: it has no model, yet each onlology and each alignment with its two aligned ontologies are consistent.

Definition 5.18 (Partial-meet revision operators for networks of ontologies). *Given a network of ontologies $\langle \Omega, \Lambda \rangle$, an ontology $O \in \Omega$ (resp. an alignment $A \in \Lambda$) and an assertion δ (resp. a correspondence μ), partial-meet revision is defined by:*

$$\langle \Omega, \Lambda \rangle \dot{\boxplus} \delta/O = \dot{\bigcap} \gamma(\langle \Omega, \Lambda \rangle \Subset \delta/O) \boxplus \delta/O$$

and

$$\langle \Omega, \Lambda \rangle \dot{\boxplus} \mu/A = \dot{\bigcap} \gamma(\langle \Omega, \Lambda \rangle \Subset \mu/A) \boxplus \mu/A$$

Partial-meet revision on networks of ontologies is indeed a valid revision operator.

Proposition 5.19 ([EUZENAT 2015]). *Partial-meet revision on networks of ontologies satisfies the six basic revision postulates for revision of networks of ontologies.*

Conclusion

Inconsistent and incoherent knowledge makes it non usable. It may be possible to repair it locally, however this becomes quickly non practical and weakly justified. Belief revision offers well-thought principles which bind the revision operators. Belief revision initially defined syntactically can better deal with ontologies when expressed semantically. Hence, this is through using the semantics defined in Chapter 2 and 3, that they are adapted to ontologies, as discussed in this course, and of alignments. Revision in networks of ontologies cannot be reduced to revising locally ontologies or alignments. Hence, operators and postulates have also been provided for networks.

With such types of operators, it is possible to ensure that the evolution of knowledge occurs consistently. Although this provides a global strategy for addressing inconsistency problems, they are usually considered within the local perspective of individual agents. This is what is considered in the following chapters.

Quiz

- What is the purpose of belief revision?
- What is the difference between theory revision and base revision?
- Which are the problems in applying belief revision to (description logic) ontologies?
- May the revision by an ontology formula lead to changes in an alignment?
- May the revision by an alignment correspondence lead to changes in an ontology?

References

[EUZENAT 2015] Jérôme Euzenat, Revision in networks of ontologies, *Artificial intelligence* 228:195-216, 2015

[FALAKH, RUDOLPH, and SAUERWALD 2023] Faiq Miftakhul Falakh, Sebastian Rudolph, and Kai Sauerwald, AGM belief revision, semantically, tech. rep. 2112.13557, arXiv, 2023

[FERMÉ and HANSSON 2018] Eduardo Fermé, Sven Ove Hansson, Belief change: introduction and overview, Springer, Cham (CH), 2018

6

Cultural knowledge evolution

In which relying on static bridges and revision operators between knowledge representations is not sufficient: common knowledge is grounded on common experience — We investigate how dynamic techniques of cultural evolution can be compatible with knowledge representation semantics — To evolve it and improve knowledge coherence.

The semantics of distributed knowledge has been provided from above, by considering an established network of ontologies. The evolution of a network of ontologies has been considered through belief revision. Belief revision also does consider the situation from above: it is based on the complete knowledge of the network. Yet, it still faces a difficulty: how to select the preferred revision?

Here the problem of knowledge evolution will be considered from an individual perspective. In general knowledge can be acquired through observing the environment. It can also through being transferred from an agent to another. However, in our societies a lot of knowledge is implicitly acquired through cooperating. Especially when agents use their knowledge to act. In particular, agents achieve it through assuming that they understand each other and when evidence is found that this is not the case, adapting their knowledge for better cooperation. This way of revising beliefs and knowledge works like if the environment, including the society of agents, applied selective pressure to agent knowledge.

I will consider this from the standpoint of co-operating agents and the influence this may have on their knowledge. In that respect, agent perception may still be considered as data in RDF and their knowledge as ontologies in OWL. Other agents are part of each agent's environments and interaction may be thought of as perceiving other agents' actions and performing actions that others will perceive. Indeed, communicating is acting [Austin 1962]. However, I will specifically distinguish between an environment and a society which is made of the agents with which an agent interacts.

Of course, it may be simpler to consider agents simply offering their knowledge to everyone. However, this would raise some problems: this knowledge may not be compatible. In order to understand each other, agents would first need to agree on common knowledge: a culture.

I introduce the perspective of cultural evolution, that I apply to knowledge (§6.1). Cultural evolution is an attempt to understand how a group of agents can reach shared knowledge through cooperating or interacting. Experimental cultural evolution provides a methodology for simulating knowledge evolution (§6.2) and this is applied to alignment repair (§6.3).

6.1 Cultural knowledge evolution: motivation

Evolution is a control mechanism based on three principles:

- Variation;
- Transmission (inheritance for biology);
- Selection.

It can be applied to a variety of objects.

Cultural evolution comes initially from anthropology; however, it has spread to the whole social science an humanities field [Plotkin 1993; Mesoudi 2011]. It is based on the study of some phenomena such as beliefs,

attitudes, knowledge, customs, values, etc. broadly called culture. These are pieces of information that may be shared between people and determine their behaviour. Culture may vary from person to person, be transmitted from person to person and thus shared among people. It is also subject to selection. Hence, it is prone to be considered under the light of evolution theory.

A specific view of evolution mechanisms is provided by the replicator-interactor model (Figure 6.1). This model considers that there is an entity called the replicator (the genome in biology) which specifies features of individuals, the interactor (the phenotype in biology). This replicator replicates (through reproduction in biology) in an imperfect manner, which causes variation, to lead to new individuals. The replicator is thus transmitted (inherited by the offspring in biology). The interactor interacts with the environment which imposes it selective pressure: by draining too much resources, it reduces reproduction. Hence, replicator transmission occurs differentiality with respect to the interactor ability to respond to selective pressure.

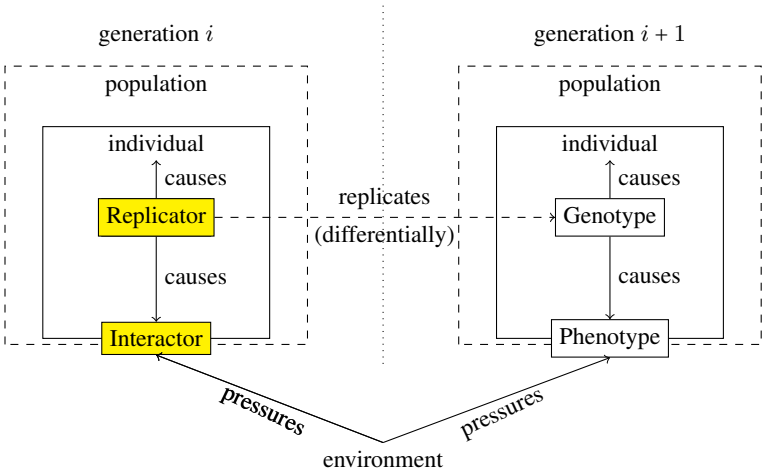


Figure 6.1. Presentation of the replicator-interactor model in biology.

Considering Figure 6.1, it is possible to distinguish between culture acting as a replicator and behaviour acting as an interactor. Indeed, culture influences behaviour but this is the behaviour which is subject to the pressure of the environment, including other people.

The main difference between cultural and biological evolution (or genetic evolution) are that:

– Culture-bearer can modify, i.e. select, their own culture.
– Transmission in not inheritance any more, it is not from parents to children.
– Culture does not create the agent (unlike replicators).

These are well-documented objections that do not prevent to call this evolutionary.

Cultural evolution can be studied in various ways [Mesoudi 2011] broadly divided into:

– by observation of actual behaviour: this is mostly what is considered in fields such as anthropology, archeology, sociology and even economics;
– by controled experiments on people or animals: this is what is achieved in psychology, ethology, etc.

However, this can also be studied by computational means. In particular, *in silico* computational experiments may be designed to study cultural evolution.

6.2 Experimental methodology for computational cultural evolution

Experimental cultural evolution applies multi-agent simulation to cultural artifacts. There has already been various such experiments for abstract culture propagation [Axelrod 1997], language transmission [Kirby, Cornish, and Smith 2008] and cultural language evolution [Steels 2012]. The latter offers a systematic experimentation framework in which agents play ‘games’.

6.2.1 Games

The game based methodology involves the following:

1. Agents with different information,
2. play a specific game or protocol,
3. with a goal to achieve, e.g. guessing something,
4. using this information;
5. at the end of each game, they adapt their information to account for the issue of the game.

The information considered here may be knowledge, know-how, rules, etc.

These types of games are relevant to cultural evolution because culture is: (*a*) information with variation (1) (*b*) that determines behaviour (in the game, 4) (*c*) can be transmitted (through jointly playing the game, 2) (*d*) and is subject to selection (by the agents 5, but motivated by their will to achieve their goal, 3).

An instance of such a game is the situated naming game [STEELS 2012] whose rules are:

1. A robot tells the name of an (randomly chosen) object (if it has no name a new one is created);
2. The other robot must identify the object and designate the object;
3. The first robot perceives what is shown and nods if it corresponds to his name (SUCCESS);
4. Otherwise (FAILURE), he points at the actual object;
5. The second robot records the outcome of the game and adapt its lexicon in consequence.

Luc Steels [STEELS 2012] distinguishes between three types of experiments:

Baseline experiments provide agents with 'perfect' information and check that agents can play the games successfully, i.e. the success rate is good. This may become a baseline for other experiments.

Acquisition experiments in which instructor agents, with 'perfect' information, are able to transmit/teach it to other agents. This shows that the perfect information can be acquired.

Formation experiments in which agents start with basic or no information and have to acquire it to play the game. So, agent can not only learn information, but also create it.

Here I concentrate on formation experiments, without targeting any 'perfect' information. Of course, in order to yield conclusive results it is necessary to check that agents improve their ability to play the game (success rate).

6.2.2 Protocol

A specific number of such games are played sequentially for a specific population of agents. Usually, at each iteration the instance of the game (agent playing, objects considered) is determined randomly. Hence, several runs of this protocol are performed.

During the game some measures are monitored. One typical measure is the success rate achieved by agents, i.e. for any iteration the proportion of previous successful games. It is important that the success rate evolves positively, because it indicates that the agents behave adequately: they modify their culture in a way that make them achieve their goals.

This approach can be defined more formally. Given a population $\mathcal{A}$ of agents and $K_{a,t}$ the cultural background of agents a at time t. For each game it is involved in, agent $a \in \mathcal{A}$ uses its culture to react to the proposed game instance s (for situation). This determines the outcome of the game $f(K_{a,t}, s)$ which may be evaluated as a success ($success(f(K_{a,t}, s))$) or failure($\neg success(f(K_{a,t}, s))$). In function of the outcome, the agent adapts its culture by applying it an adaptation operator *op*: $K_{a,t+1} = op(K_{a,t}, s, success(f(K_{a,t}, s)))$.

In the case of the naming games, the measured quantities are provided in Figure 6.2: (*a*) success rate, called communicative success, (*b*) number of concepts, prototypical views per agents, (*c*) size of lexicon used by all agents. The figure shows that (*a*) the success rate quickly reaches a value after which it always improve (always successful), (*b*) after a peak due to agents generating random names, the size of the lexicon stabilises around a size commensurate with the number of objects, (*c*) the number of prototypes agents can distinguish grows and stabilises. The confidence intervals show the stability of theses values over the several runs of the experiment.

6.2.3 Application to knowledge (and beliefs)

Such an approach can also be applied to knowledge:

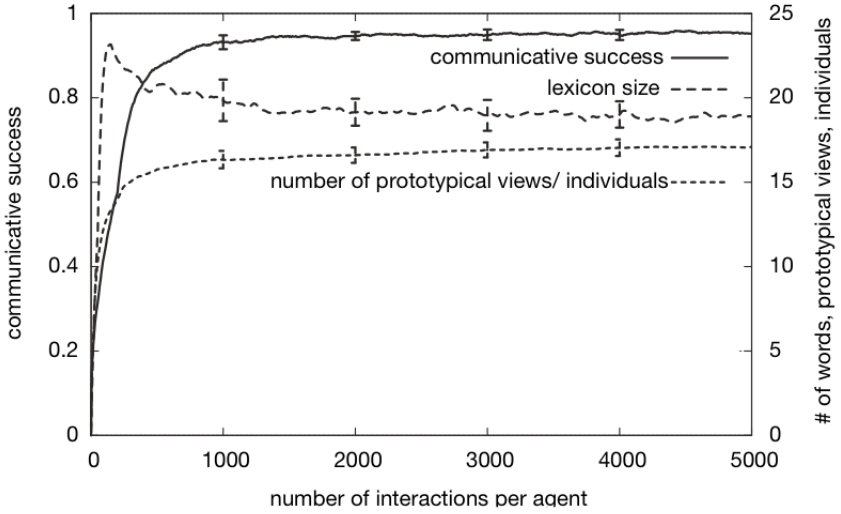


Figure 6.2. Results of the naming game (from [STEELS 2012]).

– Agents are bearing knowledge (ontologies, alignments),
– that they use for interacting with each others and their environment,
– Locally adapting their knowledge at the issue of each interaction.

Agents interpret their knowledge according to its semantics.

After running random simulation on a large number of games, it is possible to observe global properties of the resulting knowledge and of its evolution. In the case of knowledge, the most interesting properties are epistemic properties. Besides success rate there are many different knowledge-based measures that may be applied:

– consistency: is knowledge consistent, or the proportion of agents which have consistent knowledge,
– diversity: how different is each agent's knowledge?
– correctness: is knowledge correct with respect to some ideal knowledge?
– accuracy: is knowledge prone to be successful in the game?

These measures are usually not accessible to agents, because they are global (diversity), because they require external resources (correctness) or because they do not have the capabilities to compute them (consistency, see §7.6.3).

6.3 Application to cultural alignment repair

The framework of experimental cultural evolution has been applied to agent knowledge. One such application is the alignment repair game in which agents interpret their ontology and the alignments between their ontologies according to their semantics. They modify their alignments whenever the game shows them that a correspondence is not correct.

This example is covered in detail in [EUZENAT 2014a] and [EUZENAT 2017].

6.3.1 The alignment repair game

The alignment repair game (ARG) is a cultural knowledge evolution game which takes alignments as culture (not necessarily ontologies) that agents use to communicate and repair them when they reveal incorrect.

Objects are characterised by Boolean attributes which may be though of as OWL restrictions such as $\exists_{\geq 1}$ob:plays.$\top$ denoting those objects, here people, who play something, here a musical device. This may be

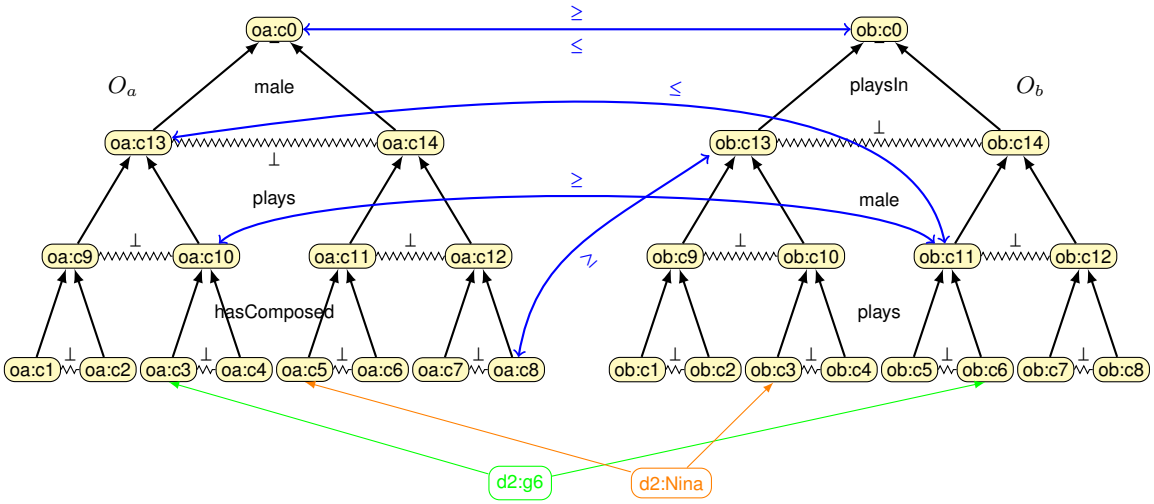


Figure 6.3. Two ontologies and one random alignment.

shortened as ob:plays, the negation ¬ob:plays would then correspond to the complementary restriction ∀ob:plays.⊥ which characterises the objects which do not play anything.

Each fully characterised object will then correspond to an object type which is a conjunction of atomic properties such as plays ∧ hasComposed ∧ playsIn ∧ ¬male, i.e. a female musician and composer who did play in a band.

Each agent is provided with an ontology shaped as a dichotomic tree using all the attributes but one in permuted orders. This means that at each level of the tree, the classes are separated into two along one single property. The ontology O_a of Figure 6.3 is defined by the sequence of properties male/plays/hasComposed which means, for instance, that class oa:c3 is defined by male ∧ ¬plays ∧ hasComposed.

Agents start with *random* alignments between their ontologies. These alignments must however satisfy two constraints: (*a*) the topmost classes of each ontology must be equivalent, and (*b*) alignments must be functional, i.e. there are no two correspondences $\langle c, \geq, c' \rangle$ and $\langle d, \geq, c' \rangle$, with $c \neq d$, between the same pair of ontologies. This starting point is illustrated by Figure 6.3.

The experimenter has access to the correct alignments between all the ontologies, that are called the reference alignments. Agents do not have access to them. They also do not have access to other agents' ontologies, only the name of their classes, which are all different.

Agents have no other goal than continuously playing an interaction game. The game is played as follows:

- Two agents (a and a') are chosen at random,
- One agent (a) picks an instance (o) at random and asks the other one (a') in which class it would classify the instance in its own ontology (O) using the public alignment A between O and O'.
- a' determines which correspondence is applicable and communicates both the correspondence ($\langle c, \geq, c' \rangle$) and the class ($c$).
- a considers the relations between c and the most specific class d in which o is classified in O ($o \sqsubset d$):
  - If c is compatible with d (concretely $d \sqsubseteq c$) the interaction is considered successful,
  - otherwise ($c \perp d$) it is a failure.
- Agent a communicate this to agent a' who uses adaptation operators to adapt the alignment.

Considering the two ontologies and the alignment of Figure 6.3, with agent a on the left and agent b on the right. The set of random correspondences is:

⟨oa:c13, ≤, ob:c11⟩ ⟨oa:c0, ≤, ob:c0⟩
⟨oa:c10, ≥, ob:c11⟩ ⟨oa:c8, ≥, ob:c13⟩ ⟨oa:c0, ≥, ob:c0⟩

If object d2:g6 *is drawn, apparently identifiable by* male ∧ ¬plays ∧ hasComposed ∧ ¬playsIn. *b will classify this object as* ¬playsIn ∧ male ∧ ¬plays, *i.e.* ob:c6. *The most specific correspondence applicable to it is* ⟨oa:c10, ≥

, ob:c11⟩. *Hence, b answers with* oa:c10. *Since a classifies* d2:g6 *as* male ∧ ¬plays ∧ hasComposed, *i.e.* oa:c3, *which is subsumed by* oa:c10, *the game is a success.*

On the contrary, if object d2:Nina *corresponding to* ¬male ∧ plays ∧ hasComposed ∧ playsIn *is drawn, then b identifies it as* playsIn ∧ ¬male ∧ plays, *i.e.* ob:c3. *The most specific applicable correspondence is* ⟨oa:c8, ≥, ob:c13⟩, *thus b answers with* oa:c8. *But a classifies* d2:Nina *as* ¬male ∧ plays ∧ hasComposed, *i.e.* oa:c5, *which is disjoint from* oa:c8, *hence this time the issue of the game is a failure.*

6.3.2 Adaptation operators

When communication fails, agents modify the alignment that has been used through adaptation operators (Figure 6.4). [EUZENAT 2017] defines six operators; we restrict ourselves to four of them.

More precisely, assuming that the correspondence $\langle c, \geq, c' \rangle$ has been used, the operators that may be considered are:

delete simply removes the correspondence;

refine extends delete by adding correspondences $\langle c, \geq, c'' \rangle$ between c and the subclasses c'' of c' that do not subsume the actual class of the object and do not break functionality, i.e. there is not another correspondence $\langle x, \geq, c'' \rangle$;

addjoin extends delete by adding a correspondence $\langle c'', \geq, c' \rangle$ between c' and the lowest superclass c'' of c compatible with d;

refadd is the combination of refine and addjoin.

Figure 6.4 illustrates these operators. addjoin, and thus refadd, requires extra steps of interaction to provide to b the subsumed classes in a ontology.

These operators share two particular properties [EUZENAT 2017]:

Safeness after applying the operator, if the same instance is chosen, the problem would not occur again (maybe a different problem would occur);

Entailment each correspondence added by any of the operators was entailed by the removed correspondence.

These two properties can be thought of as the success and inclusion postulates of a contraction operator (Definition 5.3).

Because the problem is so simple, this can be seen as:

- Revision by $\neg(o \sqsubseteq c)$ (the instance);
- Contraction by $\langle c, \geq, c' \rangle$ (the culprit).

We consider the latter because agents do not memorise data, hence they do not need to believe an assertion concerning a specific objects..

In the situation above, if after the presentation of object d2:Nina *agent b uses the* refadd *operator, it will delete the initial correspondence (*⟨oa:c8, ≥, ob:c13⟩*) and add all the green correspondences (*⟨oa:c14, ≥, ob:c13⟩, ⟨oa:c8, ≥, ob:c9⟩ *and* ⟨oa:c8, ≥, ob:c4⟩*) of Figure 6.4.*

6.3.3 Protocol

Experiments depend on various parameters (actual values in parentheses):

- the size of the population (4 agents);
- the number of features identifying objects (4 features);
- the number of games played (10000 iterations);
- the operators (delete, refine, addjoin, refadd operators);
- the number of time the experiment is repeated (10 runs).

Various measures, averaged on the number of runs, are recorded:

Success rate measures the capacity of the agents to successfully play the game. It is the ratio of success over games played.

Semantic precision and recall measures the quality of the alignments. They compute the degree of correctness and completeness of the network of ontologies with respect to the reference network (F-measure averaging them).

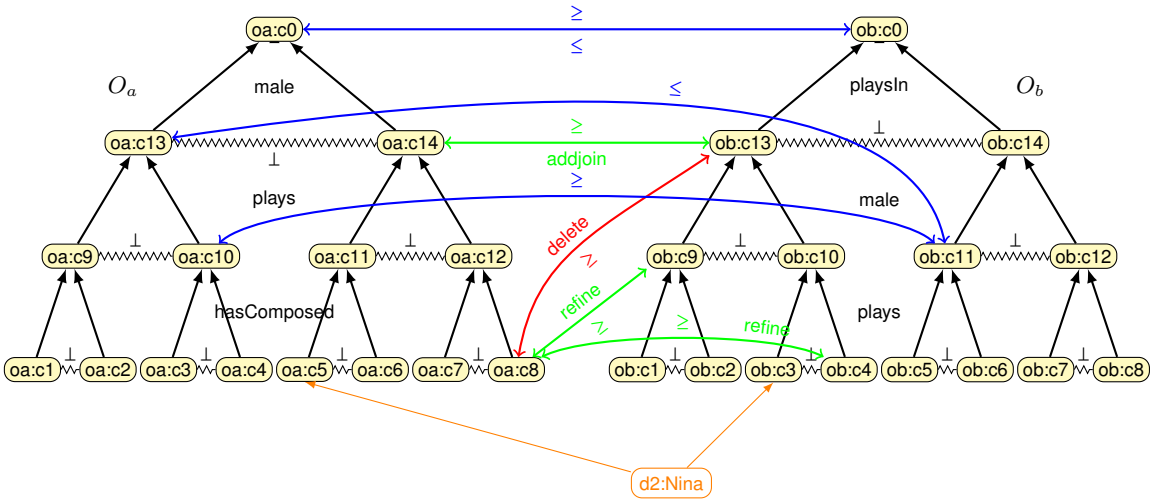


Figure 6.4. The two ontologies of Figure 6.3 and the alignment adapted by refadd.

Incoherence rate is another quality measure which does not depend on an external reference such as the ideal network. It is the proportion of incoherent correspondences in alignments.

Convergence measures the time taken by agents to converge. It is the last observed iteration at which an agent encounters a failure.

It is possible to ask the following questions:

1. Does the process converge?
2. What is the effect of adaptation operators?
3. How do they compare to baselines?

6.3.4 Results

Figure 6.5 plots the success rate for the first 2000 iterations of each runs with the addjoin operator. It shows a remarkable convergence between the runs: in spite of different random initial conditions and different games played, the success rates curves are very close. After the first 200 games dominated by randomness, the success rate converges assymptotically and at the same pace towards 100%. Indeed, as soon as the network of ontologies has been cleaned up (around 1200 iterations), it reaches a stable state and agents do not encounter any more failures. In consequence, the rate only grows. It never reaches 1 because of the initial period which contains failures.

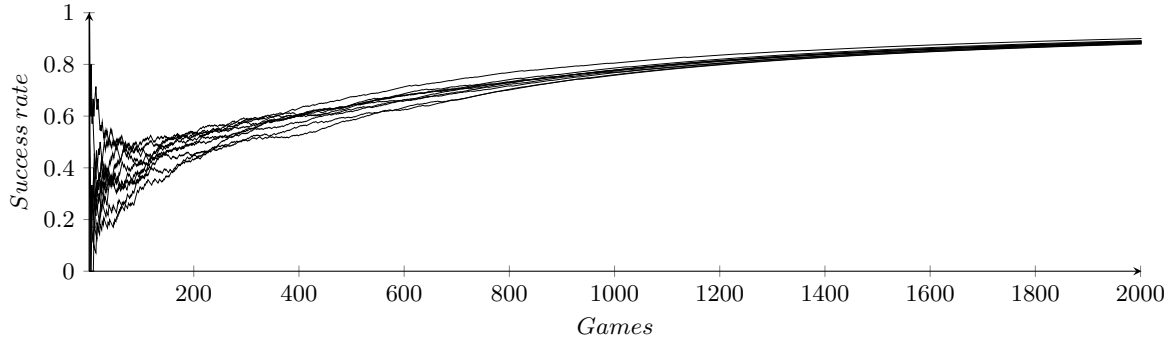


Figure 6.5. Success rate for 10 individual runs with the addjoin operator ([20140306-NOOR] operator=addjoin; #agents=4; #games=2000; #runs=10).

Figure 6.6 shows the evolution over 2000 iterations of the success rate and F-measure of the four operators. It shows that the convergence of the delete operator is achieved very quickly with the more elaborate operators converging more slowly in proportion to the number of correspondences they introduce. The quality of the delete operator is very low, barely above the initial random network of ontology. But the more elaborate the operator, the higher its quality. Actually, Table 6.1 shows that at the end of the process these operators have provided increasing recall, their capacity to entail a reference correspondence, at the expense of precision, the capacity of their correspondences to be entailed by the reference network. This quality improves with each further operator.

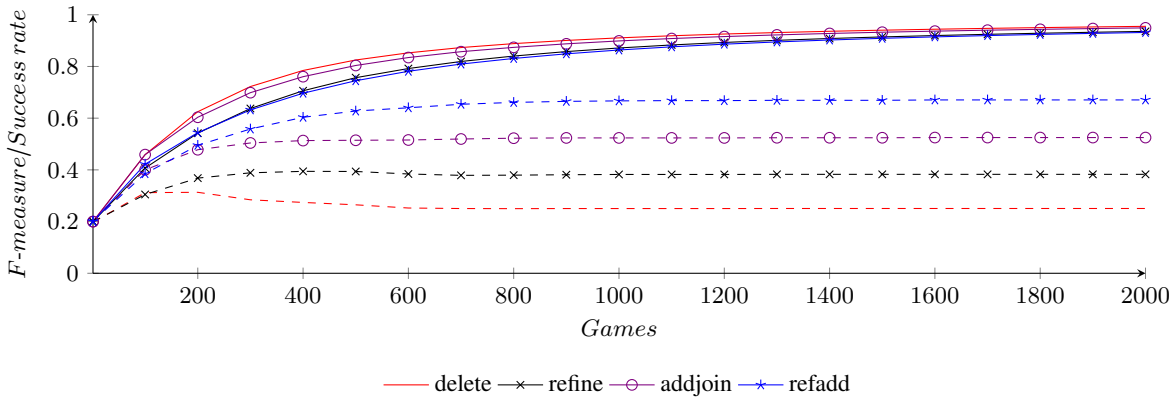


Figure 6.6. Success rate (plain) and semantic F-measure (dashed) for operators delete/refine/addjoin/refadd([20180308-NOOR] operator=delete,refine,addjoin,refadd; #agents=4; #games=2000; #runs=10).

Table 6.1 provides the measured values at the end of the 10000 iterations (averaged over the 10 runs). It shows a very high success rate: most of their life span agents are correct. They have improved on most of the measures over the initial network but semantic recall. This is due to incorrect correspondences in the initial network which entail reference correspondences but discarded because they are not themselves correct. However, their alignments are far from perfect: they miss many correct correspondences compared to the reference network and even the initial network.

In particular, Table 6.1 compare the results with those of two systems, LogMap and Alcomo, which are alignment repair systems operating a kind of belief revision in the aim of restoring consistence. The two approaches work on different bases. Alignment repair systems have knowledge of the whole network of ontologies but do not exploit the games; the agents do the opposite. Their 0 incoherence degree testifies that these systems meet their goal. Although agents do not reach full consistency, they achieve higher quality improving both precision and recall.

Operator	Size	Success rate	Inc. degree	Sem. Precision	Sem. F-measure	Sem. Recall	Max Conv.
reference	86	1.0	0.0	1.0	1.0	1.0	1
initial	54	0.24	0.34	0.11	0.20	0.89	-
delete	12	0.99	0.02	0.96	0.25	0.14	1224
refine	20	0.99	0.03	0.95	0.38	0.24	1224
addjoin	31	0.99	0.16	0.79	0.52	0.39	1526
refadd	48	0.99	0.15	0.79	0.67	0.58	1554
Alcomo	28	0.43	0.0	0.21	0.26	0.33	-
LogMap	29	0.51	0.0	0.24	0.26	0.29	-

Table 6.1. Measures at the end of the experiments and comparison with alignment repair systems ([20180308-NOOR] operator=delete,refine,addjoin,refadd; #agents=4; #games=10000; #runs=10).

6.3.5 Discussion

These results shows that, by using their knowledge in very simple games, agents are able to improve measurably and predictably the quality of their knowledge. As can be observed, agents converge to alignments which allows them to always succeed in the game. However, the quality of these alignments can be improved. The alignment repair game has been complexified by introducing modalities [EUZENAT 2017]:

relaxation to answer less precisely which allows agents to discover faultly but hidden correspondences;

generation to introduce a new relevant correspondence when none is applicable (but $\langle \top, \geq, \top \rangle$);

expansion to add new random correspondences when adaptation reduces the number of correspondences.

They have been shown to improve knowledge quality.

In addition, this cultural knowledge evolution experiments have been more broadly extended along different directions:

- Starting with empty alignments;
- Learning ontologies and alignments;
- Altering ontologies;
- Involving several populations;
- Transmitting knowledge over generations.

The same games and operators are considered theoretically in Section 7.6.

Conclusion

Agents equiped with ontologies are able to reason (following ontology semantics). In the alignment repair game, they use it to detect problems with incorrect alignment during communication. However, through co-operation and adaptation using their knowledge (game playing), they can make it evolve. This leads to the improvement of globally shared knowledge (culture).

Although the approach considered in this chapter largely differs from the previous ones, it remains very connected. This chapter did not introduce a new semantics or new objects to be given a semantics. Instead it uses the semantics of data, ontologies, alignments and eventually queries. This is what allows to detect inconsistencies between agent answers and ontologies. The inconsistency is not treated globally as would be done with belief revision, but by applying local adaptation operators, still obeying this semantics. It is the repetitive play of random events which triggers evolutionary mechanisms and improves knowledge in the long run.

Quiz

- What are the three main features of evolution theory?
- What would be the replicator and interactor in genetic? In cultural knowledge evolution?
- Why is failure important in cultural knowledge evolution games?
- How is cultural knowledge evolution related to belief revision?

References

[STEELS 2012] Luc Steels (ed.), Experiments in cultural language evolution, John Benjamins, Amsterdam (NL), 2012

[MESOUDI 2011] Alex Mesoudi, Cultural Evolution: How Darwinian theory can explain human culture and synthesize the social sciences, University of Chicago Press, Chicago (IL US), 2011

7

Multi-agent epistemic logic

IN WHICH THESE ATTEMPTS ARE RECAST IN DECADES OF TRYING TO CAPTURE KNOWLEDGE INTO LOGIC — KNOWLEDGE IS TOO WILD TO BE CAPTURED, BUT LOGIC PROVIDES A SOLID TUTOR TO GROW ALONG.

Agents may express their knowledge independently and a semantics may be given to this knowledge. It may also be connected through alignments and a semantics has been given for alignments. This is the basis for interpreting queries taking advantage of knowledge and alignments. Finally, through cooperating, agents may achieve alignments making sense with respect to the semantics.

However, this has been considered mostly from the standpoint of a single agent. It is difficult to have a global view of all agents' knowledge... and beliefs. The semantic structure of all agents, together with respect to the situation in which they are, may be considered holistically. This should allow agents to behave not just according to their own knowledge and beliefs, but to what they know and believe about about other knowledge and beliefs and their shared beliefs. From initial agents' knowledge and beliefs, it is possible to express the way agents communicate information between each others and how this transforms their beliefs. This is what multi-agent epistemic logics provide.

This chapter first progressively presents simple versions of:

– Modal logics (§7.1);
– Epistemic-doxastic logics (§7.2);
– Multi-agent epistemic-doxastic logics (§7.3);
– Multi-agent dynamic epistemic-doxastic logics (§7.5).

Before illustrating how such a multi-agent dynamic epistemic-doxastic logic may be used to model a simple cultural knowledge evolution game (§7.6).

7.1 Basic modal logic

Modalities are syntactic markers to qualify logical statements. They indicate that these have to be considered out of their immediate truth values. Historically, they were introduced for expressing that a proposition is not simply true but necessarily true.

Definition 7.1 (Syntax of modal logic). *The syntax of the basic modal logics is defined by:*

$$\begin{aligned}\phi ::= \ & p \mid \phi \wedge \psi \mid \neg\phi \\ & \mid \Box\phi\end{aligned}$$

such that $p \in \mathcal{P}$ is a proposition and $\Box$ is the necessity modality.

As usual

$$\phi \vee \psi \equiv \neg\phi \wedge \neg\psi$$
$$\phi \Rightarrow \psi \equiv \neg\phi \vee \psi$$

A dual modality, possibility, also defined as:

(possibility) $$\Diamond\phi \equiv \neg\Box\neg\phi$$

It makes sense that what is possible are those statements whose negation is not necessary.

For the sake of illustrating the concepts of modal logics, the taxonomical part of an ontology is encoded. Encoding means that the ontology axioms are translated in a way that it is possible to translate the modal formulas back into the ontology language. Consider the ontology O as displayed on the left hand side of the following table, and summarised in Figure 7.1:

Ontology O	*corresponding statements encoding (T_O)*	
o1:Person owl:disjointWith o1:Band	$\neg p \vee \neg b$	$\Box(\neg p \vee \neg b)$
o1:Person owl:disjointWith o1:Device	$\neg p \vee \neg d$	$\Box(\neg p \vee \neg d)$
o2:Ensemble rdfs:subClassOf o1:Band	$e \Rightarrow b$	$\Diamond(e \Rightarrow b)$
o2:Singer rdfs:subClassOf o1:Person	$s \Rightarrow p$	$\Diamond(s \Rightarrow p)$
o2:Singer rdfs:subClassOf o3:Voice	$s \Rightarrow v$	$\Diamond(s \Rightarrow v)$
o3:Voice rdfs:subClassOf o1:Device	$v \Rightarrow d$	$\Diamond(v \Rightarrow d)$

Finally, the encoding of O is the set of assertions of the right-hand side. The two disjointness constraints will be considered as a subtheory: $X_O = (\neg p \vee \neg b) \wedge (\neg p \vee \neg d)$. The full ontology is considered as $T_O = \Box X_O \wedge \Diamond(e \Rightarrow b) \wedge \Diamond(s \Rightarrow p) \wedge \Diamond(s \Rightarrow v) \wedge \Diamond(v \Rightarrow d)$. Hence the constraints are necessary, though the other axioms are only possible.

It is interesting to study how this information is compatible with the existence of a specific singer (s).

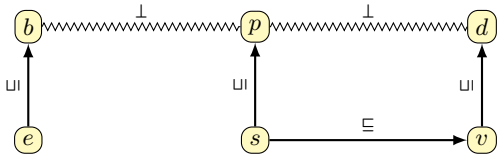


Figure 7.1. Representation of the axioms as description logic statements.

The semantics of modal logics was given with respect to a set of worlds and an accessibility relation. The structure of the relation mandates axioms of the logic.

Definition 7.2 (Kripke structure). *A* Kripke structure *is a triple $M = \langle W, \rightsquigarrow, V \rangle$ such that*

- *W is a non-empty set of worlds;*
- *$\rightsquigarrow \subseteq W \times W$ is called the accessibility relation;*
- *$V : \mathcal{P} \to 2^W$ is a propositional valuation mapping propositions to sets of worlds in which that proposition is true.*

Satisfiability in a modal logic is defined inductively with respect to a structure and a world in that structure.

Definition 7.3 (Satisfiability for modal logics). *A formula is* satisfied *by $\langle M, w \rangle$ if:*

$M, w \vDash_{ML} p$	*iff* $w \in V(p)$
$M, w \vDash_{ML} \phi \wedge \psi$	*iff* $M, w \vDash_{ML} \phi$ *and* $M, w \vDash_{ML} \psi$
$M, w \vDash_{ML} \neg\phi$	*iff* $M, w \nvDash_{ML} \phi$
$M, w \vDash_{ML} \Box\phi$	*iff* $\forall v$ *s.t.* $w \rightsquigarrow v : M, v \vDash_{ML} \phi$
$M, w \vDash_{ML} \Diamond\phi$	*iff* $\exists v$ *s.t.* $w \rightsquigarrow v : M, v \vDash_{ML} \phi$

Any non-modal formula is evaluated at a world, modal formulas depend on the related worlds.

The two structures M and M' of Figure 7.2 are two different Kripke structures. Although they cover the same set of worlds $\{w_1, w_2, w_3, w_4\}$ with the same valuation V, their accessibility relations are very different:

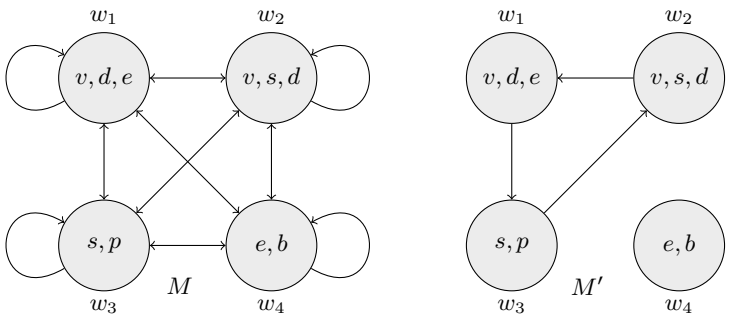


Figure 7.2. Two Kripke structures. The valuation V is displayed through the presence of propositions at each world they are true, otherwise they are false.

M is connected, reflexive, symmetric and transitive, while M' is non connected, anti-transitive, anti-reflexive, anti-symmetric, and not well-founded (contains circuits).

Both structures entail $\Box X_O$, as in no world p is true when either d or b is true. Hence, $M \models_{ML} \Box X_O$ and $M' \models_{ML} \Box X_O$.

In addition, $M \models_{ML} T_O$ because each world is accessible by all the others and there is always one world satisfying each clause. On the contrary, $M' \not\models_{ML} T_O$. More specifically, $M', w_2 \not\models_{ML} \Diamond(e \Rightarrow b)$, $M', w_3 \not\models_{ML} \Diamond(s \Rightarrow p)$ and $M', w_1 \not\models_{ML} \Diamond(s \Rightarrow v)$.

Concerning s, $M \models_{ML} \Diamond s \wedge \Diamond \neg s$. M' is very special in this case since $M', w_1 \models_{ML} \Box \neg s$, $M', w_2 \models_{ML} \Box \neg s$, $M', w_3 \models_{ML} \Box s$, hence these three connected worlds do not agree. Moreover, $M', w_4 \models_{ML} \Box(s \wedge \neg s)$. Indeed, when a world cannot access any other world, everything becomes necessary.

In the logics that will be considered here, the pair $\langle M, w \rangle$ is called a pointed model. I will adopt this vocabulary and way of doing.

For $M = \langle W, \rightsquigarrow, V \rangle$ we write:

$$M \models \phi \text{ if } \forall w \in W, M, w \models_{\text{ML}} \phi$$

but not

$$\Gamma \models_{\text{ML}} \phi \text{ if } \forall M \in \mathcal{M}(\Gamma), M \models_{\text{ML}} \phi$$

instead, we have:

$$\Gamma \models_{\text{ML}} \phi \text{ if } \forall M = \langle W, \rightsquigarrow, V \rangle, \forall w \in W, M, w \models_{\text{ML}} \Gamma \Rightarrow M, w \models_{\text{ML}} \phi$$

which is stronger.

This means that:

- a *pointed model* for a theory Γ is $\langle M, w \rangle$ such that $\forall \phi \in \Gamma$, $M, w \models_{\text{ML}} \phi$;
- a theory Γ is *satisfiable* if it has a pointed model, inconsistent otherwise;
- a formula ϕ is a *consequence* of Γ if $\Gamma \models_{\text{ML}} \phi$ (see above).

This entailment relation has the properties of a closure relation.

By imposing different properties to the accessibility relation, one obtains different logics satisfying different axioms schemata. Table 7.1 provides some of the most common axioms and the corresponding properties. For instance, axiom D requires that if something is necessary, then it may be possible. This seems obvious but given the semantics of the language it is true only if from each world, there is an accessible world.

From these axioms it is possible to define different logics which take their names from them (Table 7.2).

7.2 Epistemic-doxastic logic

An epistemic-doxastic logic aims at modelling knowledge (epistemic) and beliefs (doxastic) through modalities. It thus introduces two modalities: K for knowledge and B for belief.

axiom name	axiom scheme	property	definition
K	$\Box(\phi \Rightarrow \psi) \Rightarrow (\Box\phi \Rightarrow \Box\psi)$		
D	$\Box\phi \Rightarrow \Diamond\phi$	serial	$\forall v, \exists w; v \rightsquigarrow w$
T	$\Box\phi \Rightarrow \phi$	reflexive	$\forall v, v \rightsquigarrow v$
B	$\phi \Rightarrow \Box\Diamond\phi$	symmetric	$\forall v, \forall w, v \rightsquigarrow w \Rightarrow w \rightsquigarrow v$
4	$\Box\phi \Rightarrow \Box\Box\phi$	transitive	$\forall u, \forall v, \forall w, u \rightsquigarrow w \wedge w \rightsquigarrow v \Rightarrow u \rightsquigarrow v$
5	$\Diamond\phi \Rightarrow \Box\Diamond\phi$	euclidean	$\forall u, \forall v, \forall w, u \rightsquigarrow w \wedge u \rightsquigarrow v \Rightarrow v \rightsquigarrow w$

Table 7.1. Axiom schemata for modal logics.

logic name	base axioms	satisfied axioms	relation type
K		K	any
KT	KT	KTD	
KD45	KD45	KD45	
S4	KT4	KDT4	
S5	KT5	KDTB45	equivalence relation

Table 7.2. Modal logics defined from their axioms.

Definition 7.4 (Syntax of epistemic-doxastic logic). *The syntax of the epistemic-doxastic logic is defined by:*

$$\begin{aligned}\phi ::= \; & p \mid \phi \wedge \psi \mid \neg\phi \\ & \mid K\phi \mid B\phi\end{aligned}$$

such that $p \in \mathcal{P}$ is a proposition and K and B are the knowledge and belief modalities.

As usual, dual modalities are also defined as:

$$\hat{K}\phi \equiv \neg K\neg\phi$$
$$\hat{B}\phi \equiv \neg B\neg\phi$$

Axiom schemata of modal logics may be reinterpreted with respect to the epistemic and doxastic modalities:

axiom name	axiom scheme	interpretation
K	$K(\phi \Rightarrow \psi) \Rightarrow (K\phi \Rightarrow K\psi)$	distributivity
D	$K\phi \Rightarrow \neg K\neg\phi$	consistency
N	$K\top(\equiv K(\phi \vee \neg\phi))$	necessity
T	$K\phi \Rightarrow \phi$	truth
B	$\phi \Rightarrow K\neg K\neg\phi$	
4	$K\phi \Rightarrow KK\phi$	positive introspection
5	$\neg K\neg\phi \Rightarrow K\neg K\neg\phi$	negative introspection

Table 7.3. Axiom schemata reinterpreted for epistemic logics.

Usually, K is a S5 modality and B a KD45 modality. This corresponds to the 'Knowledge is true belief' definition: then (T): $\Box\phi \Rightarrow \phi$ holds for K, but not for B.

Additional glue axioms may be defined which relate the two modalities:

$$(\mathrm{KB}_1) \qquad K\phi \Rightarrow B\phi$$
$$(\mathrm{KB}_2) \qquad B\phi \Rightarrow KB\phi$$

The semantics of epistemic-doxastic logics uses the same set of worlds in which an agent may be situated, but two different accessibility relations. These two relations will have different structures and to satisfy the different axioms of the modalities.

We use the Epistemic-doxastic structure of [BALTAG and SMETS 2008] which has the advantage of using a single relation from which the two specific relations may be derived systematically.

Definition 7.5 (Epistemic-doxastic structure). *An* epistemic-doxastic structure *is a triple* $M = \langle W, \geq, V \rangle$ *such that*

– W *is a non-empty set of worlds;*
– $\geq \subseteq W \times W$ *is a converse well-founded, locally connected, preorder*[1] *on* W *(plausibility relation);*
– $V : \mathcal{P} \to 2^W$ *is a propositional valuation mapping propositions to sets of worlds in which that proposition is true.*

The plausibility relation $w \geq v$ reads as 'v is at least as plausible as w' (mind the inversion of v and w). In figures, the arrows point to the most plausible world. Hence, if $w \geq v$, then v is most plausible and the arrow points to v.

This plausibility relation covers both an accessibility relation to be used for knowledge and a plausibility relation to be used for beliefs. The epistemic and doxastic relations are defined on W accordingly:

$$w \sim v \text{ if } w \, (\leq \cup \geq)^* \, v$$
$$w \rightsquigarrow v \text{ if } v \in Max_{\leq}\{v \in W \mid w \sim v\}$$

This particular definition makes that $\sim$ is an equivalence relation, as required for S5 supposed to represent knowledge. The equivalence class of a world w for this relation $|w| = \{v \in W \mid w \sim v\}$ is called the information cell of w. Each epistemic assertion is thus interpreted within the information cell of the considered world and it has the same interpretation for each world of the cell. In addition, $\rightsquigarrow$ returns the most plausible worlds in the information cell of w. The fact that beliefs are evaluated within the information cell directly entails the satisfaction of Axiom KB_1. KB_2 is also satisfied because each world in the information cell has the same direct most plausible world through $\rightsquigarrow$.

Definition 7.6 (Satisfiability for epistemic-doxastic logic). *A formula is* satisfied *by a pointed model* $\langle M, w \rangle$ *if:*

$$M, w \vDash_{EL} K\phi \qquad \textit{iff } \forall v \textit{ s.t. } w \sim v : M, v \vDash_{EL} \phi$$
$$M, w \vDash_{EL} B\phi \qquad \textit{iff } \forall v \textit{ s.t. } w \rightsquigarrow v : M, v \vDash_{EL} \phi$$

This definition and the following ones are not complete. They only display how to interpret what is new. They should be completed by the same inductive decomposition as in the previous definitions, i.e. Definition 7.3.

The previous example may be encoded with epistemic and doxastic modalities. More precisely, the constraints are expressed as knowledge and the other clauses as beliefs.

The structures of Figure 7.3 have the same epistemic relation which identifies two information cells: $|w_1| = |w_2| = |w_3|$ *and* $|w_4|$*. Knowledge and beliefs are evaluated within these cells.*

Knowledge is evaluated in all worlds of an information cell and $M \vDash_{EL} KX_O$ *and* $M' \vDash_{EL} KX_O$ *as the constraints are still satisfied in all worlds. Beliefs are evaluated in the maximal elements of each cell. For each cell, the maximum worlds for the doxastic relation (identified in green) may be different leading to different beliefs. For instance,* $M \vDash_{EL} B(s \Rightarrow v)$ *because* $(s \Rightarrow v)$ *holds in both* w_2 *and* w_4*. This is not the case of* M' *because* $M', w_3 \nvDash_{EL} B(s \Rightarrow v)$*. Conversely,* $M' \vDash_{EL} B(s \Rightarrow p)$ *but not* M *because* $M, w_2 \nvDash_{EL} B(s \Rightarrow p)$*.*

7.3 Multi-agent epistemic-doxastic logics

So far the logic has been considered from the standpoint of the knowledge of a single agent. It is far more interesting to see it through several interacting agents.

The jump from epistemic logic to multi-agent epistemic logic is simple. Multi-agent epistemic logics:
– use as many pairs of modalities $\langle B_a, K_a \rangle$ as there are agents $a \in A$.

[1] The relation satisfies:
converse well-founded $\forall S \subseteq W, S \neq \varnothing, \{w \in S | \forall v \in S, w \geq v \Rightarrow w = v\} \neq \varnothing$
locally connected $\forall v, w \in W, \langle v, w \rangle \in (\geq \cup \geq^{-1})^* \Rightarrow v \geq w$ or $w \geq v$
reflexive and transitive $\forall u, v, w \in W, v \geq v$ and $u \geq v \wedge v \geq w \Rightarrow u \geq w$

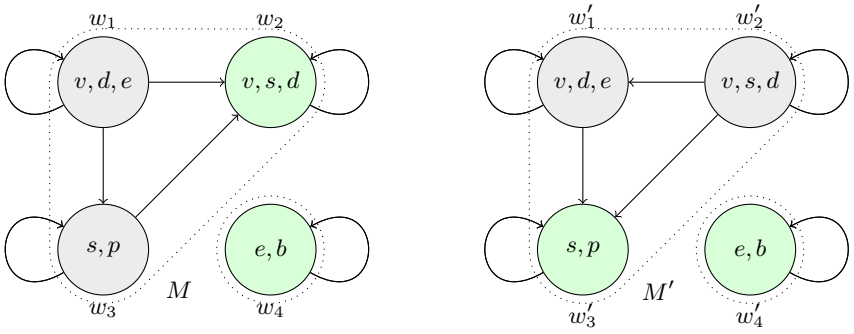


Figure 7.3. Two epistemic-doxastic structures. Surrounded by dotted lines are the information cells (conected components) and, in green, their maximal elements.

– and as many accessibility relations $\langle \sim_a, \rightsquigarrow_a \rangle$.
– A (pointed) model should satisfy the axioms of all agents.

Definition 7.7 (Syntax of multi-agent epistemic-doxastic logic). *The syntax of the multi-agent epistemic-doxastic logic is defined by:*

$$\phi ::= p \mid \phi \wedge \psi \mid \neg\phi$$
$$\mid K_a\phi \mid B_a\phi$$

such that $p \in \mathcal{P}$ is a proposition and K_a and B_a are the knowledge and belief modalities for each agent $a \in \mathcal{A}$.

Dual modalities are also defined as:

$$\hat{K}_a\phi \equiv \neg K_a \neg\phi$$
$$\hat{B}_a\phi \equiv \neg B_a \neg\phi$$

Definition 7.8 (Multi-agent epistemic-doxastic structure). *A* multi-agent epistemic-doxastic structure *is a triple $M = \langle W, \{\geq_a\}_{a\in\mathcal{A}}, V\rangle$ such that*
– *W is a non-empty set of worlds;*
– *$\{\geq_a\}_{a\in\mathcal{A}} \subseteq W \times W$ are converse well-founded, locally connected preorder on W (plausibility relation), one for each agent;*
– *$V : \mathcal{P} \to 2^W$ is a propositional valuation mapping propositions to sets of worlds in which that proposition is true.*

The ontology of the example, may be expressed with respect to two different agents a and b. Both knowledge and beliefs may be reformulated with the corresponding operators. Figure 7.4 shows a multi-agent structure for this example and the derived epistemic ($\sim$) and doxastic ($\rightsquigarrow$) relations for agents a and b.

Contrary to previously, the agents have a different epistemic structure. Agent a has the same epistemic-doxastic strutures as M, through agent b has three cells. It however remains true that they entail the constraints expressed as knowledge ($M \vDash_{MAEL} K_aX_O \wedge K_bX_O$).

On the doxastic side, according to M, $M \vDash_{MAEL} B_a(e \Rightarrow b) \wedge B_a(s \Rightarrow v) \wedge B_a(v \Rightarrow d)$ but $M \nvDash_{MAEL} B_a(s \Rightarrow p)$ because $M, w_2 \nvDash_{MAEL} B_a(s \Rightarrow p)$ and $M \vDash_{MAEL} B_b(s \Rightarrow p) \wedge B_b(v \Rightarrow d)$ only because $M, w_1 \nvDash_{MAEL} B_b(e \Rightarrow b)$ and $M, w_3 \nvDash_{MAEL} B_b(s \Rightarrow v)$.

$M \vDash_{MAEL} B_a\neg p$ and there is no atomic proposition that b can believe in this model.

The plausibility relation $w \geq_a v$ reads as 'w is at least as plausible as v for agent a' and the epistemic and doxastic relations are defined on W accordingly:

$$w \sim_a v \text{ if } w \, (\leq_a \cup \geq_a)^* \, v$$
$$w \rightsquigarrow_a v \text{ if } v \in Max_{\leq_a}\{v \in W \mid w \sim_a v\}$$

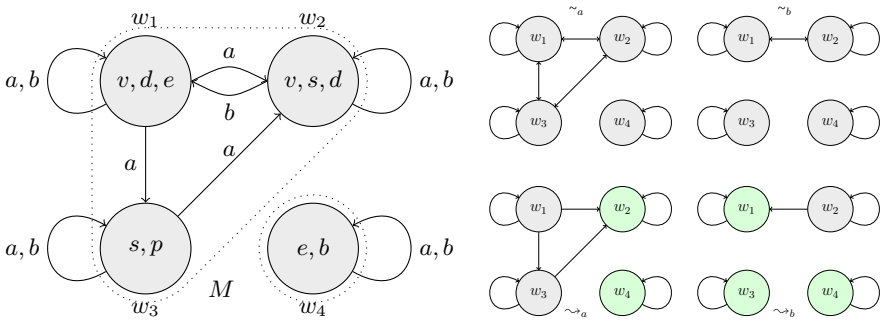


Figure 7.4. A multi-agent epistemic structure with associated epistemic ($\sim$) and doxastic ($\rightsquigarrow$) relations.

Definition 7.9 (Satisfiability for multi-agent epistemic-doxastic logic). *A formula is* satisfied *by a pointed model $\langle M, w\rangle$ if:*

$$M, w \vDash_{MAEL} K_a\phi \qquad \text{iff } \forall v \text{ s.t. } w \sim_a v : M, v \vDash_{MAEL} \phi$$
$$M, w \vDash_{MAEL} B_a\phi \qquad \text{iff } \forall v \text{ s.t. } w \rightsquigarrow_a v : M, v \vDash_{MAEL} \phi$$

This allows to express statements such as 'Bob knows that Alice believes that an ensemble is a band' ($M \vDash_{MAEL} K_b B_a(e \Rightarrow b)$) which is true in all worlds $w \in W$ such that $\forall w'; w \sim_b w'$, $\forall w''; w' \rightsquigarrow_a w''$, $M, w'' \vDash e \Rightarrow b$. This is the case as, from any worlds accessible to b, the beliefs of a are reduced to those of w_2 or w_4 in which this statement is true.

'Alice believes that Bob does not know that an ensemble is a band' ($M \vDash_{MAEL} B_a\neg K_b(e \Rightarrow b)$) is true only in w_1, w_2 and w_3. This is because it is interpreted as true in pointed models $\langle M, w\rangle$ such that $\forall w' \in W; w \rightsquigarrow_a w', \exists w'' \in W; w' \sim_b w''$ and $M, w'' \vDash e \wedge \neg b$. For a there are two information cells: $\{w_1, w_2, w_3\}$ whose maximal element is w_2 and $\{w_4\}$. In the former case, w_2 give access to w_1 and w_2 and $M, w_1 \vDash e \wedge \neg b$. Thus, $M, w \vDash_{MAEL} B_a\neg K_b(e \Rightarrow b)$ for $w \in \{w_1, w_2, w_3\}$. In the latter case, w_4 give access to itself in which $e \wedge \neg b$ is not true. Hence, $M, w_4 \nvDash_{MAEL} B_a\neg K_b(e \Rightarrow b)$.

7.4 Distributed and common belief and knowledge

Knowledge and belief modalities (K_a and B_a) marks what agent $a \in \mathcal{A}$ knows and believes. But what does a group of agents $G \subseteq A$ knows collectively? What do they believe? What are the relation between both? (do KB_1 and KB_2 hold for these?)

There is a hierarchy of answers for knowledge[2] to this question:

$CK_G\phi$ Common knowledge: Not only everybody knows ϕ, but everybody knows that everybody knows ϕ, etc. ($CK_G\phi \Leftrightarrow (\phi \wedge EK_GCK_G\phi)$);

$EK_G\phi$ Everybody knows: $\forall a \in G; K_a\phi \simeq \bigwedge_{a\in G} K_a\phi$;

$SK_G\phi$ Someone knows: $\exists a \in G; K_a\phi \simeq \bigvee_{a\in G} K_a\phi$;

$DK_G\phi$ Distributed knowledge: $\cup_{a\in A}\{\psi \mid \vDash K_a\psi\} \vDash \phi$

ϕ Aware: ϕ is true (or $\neg\phi$) but no agent knows it.

This hierarchy is organised from the stronger (top) to the weaker (bottom): if a statement satisfy some level, then it satisfies the level below.

A precise semantics can be given to these different modalities:

Proposition 7.10 (Semantics of common and distributed knowledge [FAGIN, HALPERN, MOSES, and VARDI 1995]). *Given a set of agents $G \subseteq A$ and a pointed model $\langle M, w\rangle$:*

[2] Inspired from https://blog.acolyer.org/2015/02/16/knowledge-and-common-knowledge-in-a-distributed-environment/.

$$
\begin{aligned}
M, w \vDash_{MAEL} CK_G\phi \quad & \text{iff } \forall v \text{ s.t. } w \sim^*_{\exists G} v : M, v \vDash_{MAEL} \phi \\
M, w \vDash_{MAEL} EK_G\phi \quad & \text{iff } \forall a \in G : M, v \vDash_{MAEL} K_a\phi \\
M, w \vDash_{MAEL} SK_G\phi \quad & \text{iff } \exists a \in G : M, v \vDash_{MAEL} K_a\phi \\
M, w \vDash_{MAEL} DK_G\phi \quad & \text{iff } \forall v \text{ s.t. } w \sim_{\forall G} v : M, v \vDash_{MAEL} \phi
\end{aligned}
$$

with $\sim_{\forall G} = \bigcap_{a\in G} \sim_a$, $\sim_{\exists G} = \bigcup_{a\in G} \sim_a$ *and* $\sim^*$ *the transitive closure of* $\sim$.

The same can be defined for beliefs.

So far we have seen from the previous section that the constraints are known by everyone ($M \vDash_{MAEL} EK_{a,b}X_O$)*, hence they are also believed by every one* ($M \vDash_{MAEL} EB_{a,b}X_O$)*. But, according to Propostion 7.10, they are also common knowledge* ($M \vDash_{MAEL} CK_{a,b}X_O$)*.*

It is also possible to see that $v \Rightarrow d$ *is believed, but not known, by everyone* $M \vDash_{MAEL} EB_{a,b}(v \Rightarrow d)$ *and* $M \nvDash_{MAEL} EK_{a,b}(v \Rightarrow d)$.

Finally, each statement of the whole ontology is believed by someone ($M \vDash_{MAEL} SB_{a,b}(e \Rightarrow b) \wedge SB_{a,b}(s \Rightarrow p) \wedge SB_{a,b}(s \Rightarrow v) \wedge SB_{a,b}(v \Rightarrow d)$) *because as seen before* $M \vDash_{MAEL} B_a(e \Rightarrow b) \wedge B_a(s \Rightarrow v) \wedge B_{a,b}(v \Rightarrow d) \wedge B_b(s \Rightarrow p)$.

Then $M \nvDash_{MAEL} DB_{a,b}s \vee DB_{a,b}\neg s$ *because, depending of the worlds, the beliefs of agents* a *and* b *are different.*

7.5 Communication as action: dynamic epistemic logic

Thus far, agents hold belief and knowledge, but they do not talk much. It is useful that agents can communicate their knowledge or belief. Dynamic logics deal with action, i.e. world transformation. This is appropriate because communication is action [Austin 1962]. In dynamic epistemic logics, the communicative actions are modelled as modalities which act as model transformers. This is different from other kinds of modalities that are used to navigate in the model structure and evaluate propositions with respect to the position in the structure. Transformational modalities modify the structure from which a proposition is evaluated.

Here, we mostly deal with 'Public announcement logic' in which the communicative actions are do not need an actor and are available to all agents. More appropriate, but more complex, are private announcements which are only available to a subset of the agents.

The syntax of dynamic epistemic logic defines such operators as modalities:

Definition 7.11 (Syntax of DEL). *The syntax of multi-agent dynamic epistemic logic is defined by:*

$$
\begin{aligned}
\phi ::= p \mid \phi \wedge \psi \mid \neg\phi \mid K_a\phi \mid B_a\phi \\
\mid [!\phi]\psi \mid [\uparrow\phi]\psi
\end{aligned}
$$

such that $p \in \mathcal{P}$ *is a proposition,* $a \in \mathcal{A}$ *is an agent and* $[!\phi]$ *and* $[\uparrow\phi]$ *are dynamic upgrades.*

Models are multi-agent epistemic-doxastic structures (Definition 7.8).

Definition 7.12 (Satisfiability for DEL). *A formula is* satisfied *by a pointed model* $\langle M, w\rangle$ *if:*

$$
\begin{aligned}
M, w \vDash_{DEL} [!\phi]\psi \quad & \text{iff } M^{!\phi}, w \vDash_{DEL} \psi \\
M, w \vDash_{DEL} [\uparrow\phi]\psi \quad & \text{iff } M^{\uparrow\phi}, w \vDash_{DEL} \psi
\end{aligned}
$$

where $!\phi$ *and* $\uparrow\phi$ *are model transformers* $!\phi : M \to M^{!\phi}$ *and* $\uparrow\phi : M \to M^{\uparrow\phi}$ *with* $\|\phi\|_M = \{w \in W \mid M, w \vDash_{DEL} \phi\}$ *such that*

Announcement *(*$!\phi$*) Delete all '*$\neg\phi$*'-worlds from the model. I.e.* $W^{!\phi} = \|\phi\|_M$, $w \geq_a^{!\phi} v$ *iff* $w \geq_a v$ *and* $w, v \in W^{!\phi}$, $V^{!\phi}(p) = V(p) \cap \|\phi\|_M$*;*

Conservative upgrade *(*$\uparrow\phi$*) Change the plausibility orders so that the best '*ϕ*'-worlds become better than all other worlds, while the old ordering on the rest of the worlds remains. I.e.* $W^{\uparrow\phi} = W$, $w \geq_a^{\uparrow\phi} v$ *iff either* $w \in Max_{\leq_a}(|v|_a \cap \|\phi\|_M)$ *or* $w \geq_a v$, $V^{\uparrow\phi}(p) = V(p)$.

Dynamic upgrades are announcement of knowledge and beliefs that agents must hold. They are a way to trigger and to implement belief revision through dynamic operators.

Dynamic epistemic logics use the same semantic structures as multi-agent epistemic logics. Only the effect of dynamic modalities have to be considered. Figure 7.5 shows the structure M of Figure 7.4 after conservatively upgrading by s or announcing s.

Conservative upgrades do not affect knowledge: they change the plausibility order but not its connectivity. Hence, $M \vDash_{DEL} [\uparrow s](K_a X_O \wedge K_b X_O)$ (or $M^{\uparrow s} \vDash_{MAEL} K_a X_O \wedge K_b X_O$), as before. It does, however, affect beliefs through the change of the plausibility relation. Here, its effect has been to change $w_1 \geq_b w_2$ into $w_2 \geq_b^{\uparrow s} w_1$. Thus, after the upgrade, the maximal world of the information cell of w_1 has become w_2 for b. No change has affected a's beliefs (the plausibility relation has not changed), but $M \vDash_{DEL} [\uparrow s]B_b(e \Rightarrow b)$ and $M \nvDash_{DEL} [\uparrow s]B_b(s \Rightarrow p)$ because $M, w_2 \nvDash_{MAEL} B_b(s \Rightarrow p)$.

It can be expected that, by promoting worlds with s, the upgrade would lead a and b to believe it. This is not the case because the upgrade does not change the information cells, only the order between worlds within a cell. Hence agents still do not believe s as the singleton cell made of world w_4 does not entail it.

Similarly, announcement, by reducing the set of worlds should not decrease knowledge, but in general it increases it. Hence, $M \vDash_{DEL} [!s](K_a X_O \wedge K_b X_O)$. Moreover, suppressing all worlds not entailing s, makes s known from a and b ($M \vDash_{DEL} [!s](K_a s \wedge K_b s)$).

Concerning beliefs, the announcement has not changed those of a concerning $s \Rightarrow p$ since there already was $M, w_2 \nvDash_{MAEL} B_a(s \Rightarrow p)$. But the beliefs of b in $M^{!s}$ have changed, since w_2 is now the maximum element of one of its cell: it also entails that b does not believe $s \Rightarrow p$ any more ($M \nvDash_{DEL} [!s]B_b(s \Rightarrow p)$). On the contrary, because world w_1 has been suppressed (and is not the maximum element of a cell), this leads to $M \vDash_{DEL} \neg B_b(e \Rightarrow v) \wedge [!s]B_b(e \Rightarrow v)$.

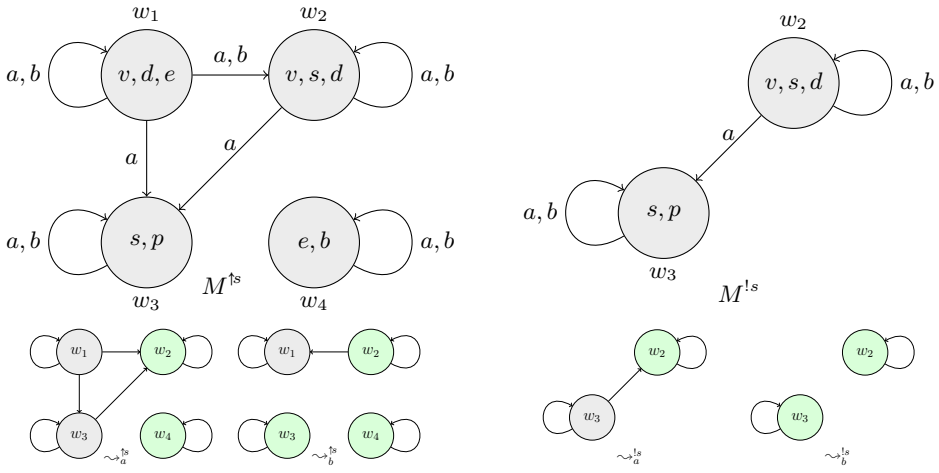


Figure 7.5. The (Dynamic) multi-agent epistemic structures obtained from that of Figure 7.4 after the conservative upgrade of s (left) and the announcement of s (right). At the bottom are the corresponding doxastic relations.

Such models provide a way to consider how different agents may see the world. It is thus possible to determines which 'moves', e.g. knowledge adaptation, are reasonable for the agents.

7.6 Modelling the alignment repair game

Dynamic epistemic logic may be used in order to model the alignment repair game (§6.3.1) in logic and to assess the properties of the game [VAN DEN BERG, ATENCIA, and EUZENAT 2021]. For that purpose, translation operations are provided such that (see Figure 7.6):

– Agent's ontologies are modelled as knowledge;

– Alignments are modelled as beliefs;
– The communication between agents is modelled through communicative modalities;
– As well as adaptation operators.

The state of the ARG system at a particular iteration is a set of ontologies held by agents and a set of alignments between them. It can thus be considered as a network of ontology (§3.2), that are called ARG states here.

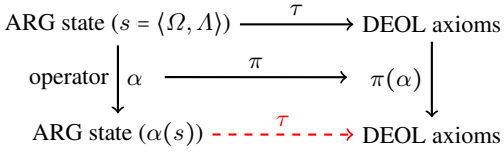


Figure 7.6. The translation from ARG to dynamic epistemic logic.

The correspondence between the actions performed by the agents in the game and what holds in its logical translation can be considered through two properties:

(correctness) $\forall s, (\tau(s))^{\pi(\alpha)} \vDash_{\text{DEOL}} \tau(\alpha(s))$

(completeness) $\forall s, \tau(\alpha(s)) \vDash_{\text{DEOL}} (\tau(s))^{\pi(\alpha)}$

Instead of asserting s, one would assert $\mathsf{o2{:}Singer}(\mathsf{d2{:}Nina})$.

7.6.1 DEOL: Dynamic epistemic ontology logic

An extension of multi-agent dynamic epistemic logic, called DEOL, is introduced to express networks of ontologies (ARG states). It allows to encode, in a more elegant way than that given in Section 7.1 (p.78), the ontologies and alignments used in the game. They only use class subsumption ($\sqsubseteq$) and disjointness ($\perp$) statements as well as class membership statements.

Definition 7.13 (Syntax of DEOL). *The syntax of multi-agent dynamic epistemic ontology logic is defined by:*

$$\phi ::= \phi \wedge \psi \mid \neg\phi \mid K_a\phi \mid B_a\phi \mid [!\phi]\psi \mid [\uparrow\phi]\psi$$
$$\mid C(o) \mid C \sqsubseteq D \mid C \perp D$$

such that C and D are class names and o an object name.

Here, it is implicit that $C(o)$ corresponds to $\mathsf{rdf{:}type}(o, C)$, $C \sqsubseteq D$ to $\mathsf{rdfs{:}subClassOf}(C, D)$ or to the correspondence $\langle C, \leq, D \rangle$, $C \perp D$ to $\mathsf{owl{:}disjointWith}(C, D)$. Hence, instead of writing $s \Rightarrow p$ as in the previous section, one will write $\mathsf{Singer} \sqsubseteq \mathsf{Person}$. The novelty is that individual objects can be assigned to classes.

DEOL structures extend DEL structures in order to interpret ontological statements with respect to OWL interpretations (§2.3.2). Hence, the main change is the use of OWL-like interpretations instead of propositional valuations.

Definition 7.14 (DEOL structure). *A* DEOL structure *is a quadruple $M = \langle W, \{\geq_a\}_{a \in \mathcal{A}}, \Delta, \{I_w\}_{w \in W} \rangle$ such that*

– *W is a non-empty set of worlds;*
– *$\{\geq_a\}_{a \in \mathcal{A}} \subseteq W \times W$ are the plausibility relations on W, one for each agent, that are converse well-founded, locally connected preorders;*
– *Δ is the domain of interpretation;*
– *$\{I_w\}_{w \in W}$ are OWL interpretations of the vocabulary used in the worlds of W.*

Satisfiability in a pointed model is defined exactly as it would for OWL.

Definition 7.15 (Satisfiability for DEOL). *A formula is* satisfied *by a pointed model $\langle M, w\rangle$ if:*

$$\begin{aligned}
M, w \models_{DEOL} C(o) &\quad \text{iff } I_w \models_{OWL} \mathsf{rdf{:}type}(o, C)\\
M, w \models_{DEOL} C \sqsubseteq D &\quad \text{iff } I_w \models_{OWL} \mathsf{rdfs{:}subClassOf}(C, D)\\
M, w \models_{DEOL} C \sqsubseteq C' &\quad \text{iff } I_w, I_{w'} \models_{\Delta} \langle C, \leq, C'\rangle\\
M, w \models_{DEOL} C \bot D &\quad \text{iff } I_w \models_{OWL} \mathsf{owl{:}disjointWith}(C, D)
\end{aligned}$$

Other formulas are interpreted like in Definition 7.12.

7.6.2 Translations

Once DEOL is defined, it is necessary to translate ARG states into a set of DEOL axioms. This is achieved with a function (τ) translating ontological statements as knowledge of the agent holding the ontology and correspondences as beliefs shared by agents holding the related ontologies:

Definition 7.16 (Translation τ). *The translation τ from ARG states to DEOL theories is defined by:*

$$\tau(\langle\{O_a\}_{a\in\mathcal{A}}, \{A_{ab}\}_{a,b\in\mathcal{A}}\rangle) = \bigcup_{a\in\mathcal{A}}\{K_a(\phi)|\phi \in O_a\} \cup \bigcup_{a,b\in\mathcal{A}}\{B_a(\gamma) \wedge B_b(\gamma)|\gamma \in A_{ab}\}$$

The translation provided by τ is faithful [VAN DEN BERG, ATENCIA, and EUZENAT 2021] in the sense that:

Proposition 7.17 (Translation faithfulness [VAN DEN BERG, ATENCIA, and EUZENAT 2021]). *Let s be an ARG state for a set of agents $\mathcal{A}$, then for each agent $a \in \mathcal{A}$,*

(ontology consequence preservation) $\quad \forall\phi, O_a \models_{OWL} \phi \Rightarrow \tau(s) \models_{DEOL} K_a\phi$

(alignment consequence preservation) $\quad \forall\gamma, A_{ab} \models_{\Delta} \gamma \Rightarrow \tau(s) \models_{DEOL} B_a\gamma$

if, in addition, s is locally consistent

(strict knowledge adherence) $\quad \forall\phi, \tau(s) \models_{DEOL} K_a\phi \Rightarrow O_a \models_{OWL} \phi$

Belief adherence does not hold because logical agents are able to reason globally, i.e. to compose their beliefs, though ARG agents only reason locally with their ontology and one alignment at a time.

Then, the outcome of a game is rendered by announcements. There is a public announcement for the answer of the queried agent and, if necessary, a set of conservative upgrades for the adaptation operator.

Definition 7.18 (ARG Dynamics in DEOL). *Let $T = \tau(s)$ be the DEOL theory that is the translation of an ARG state s. Then $\pi(\alpha[\langle C_a, \geq, C_b\rangle, o]) : T \mapsto T'$ is a theory transformation, with adaptation operator α, correspondence $\langle C_a, \geq, C_b\rangle$ and object o, where T' is defined as:*

$$T' := \begin{cases} T \cup \{[!C_b(o)]\top\} & \text{if } O_a \models_{DEOL} C_a(o)\\ T \cup \{[!C_b(o)]\top, \left[\pi(\alpha[\langle C_a, \geq, C_b\rangle, o])\right]\top\} & \text{if } O_a \not\models_{DEOL} C_a(o)\end{cases}$$

such that $\pi(\alpha[\langle C_a, \geq, C_b\rangle, o])$ is the translation of the adaptation operator α applied to $\langle C_a, \geq, C_b\rangle$ with object o.

Definition 7.19 (Adaptation operators as dynamic upgrades). *Let a correspondence $\langle C_a, \geq, C_b\rangle$, an object o, and an adaptation operator α, then $\pi(\alpha)$ is defined as:*

$$\begin{aligned}
\pi(\mathsf{delete}[\langle C_a, \geq, C_b\rangle, o]) &= \uparrow(\neg(C_a \sqsupseteq C_b))\\
\pi(\mathsf{addjoin}[\langle C_a, \geq, C_b\rangle, o]) &= \uparrow(\neg(C_a \sqsupseteq C_b) \wedge mscc_a(o, C_a) \sqsupseteq C_b)\\
\pi(\mathsf{refine}[\langle C_a, \geq, C_b\rangle, o]) &= \uparrow(\neg(C_a \sqsupseteq C_b) \wedge \bigwedge_{C'_b\in M_b(o,C_b)} C_a \sqsupseteq C'_b)\\
\pi(\mathsf{refadd}[\langle C_a, \geq, C_b\rangle, o]) &= \uparrow(\neg(C_a \sqsupseteq C_b) \wedge mscc_a(o, C_a) \sqsupseteq C_b \wedge \bigwedge_{C'_b\in M_b(o,C_b)} C_a \sqsupseteq C'_b)
\end{aligned}$$

such that $mssca(o, C) = \mu_{\sqsubseteq}\{D \in O; O \vDash_{OWL} D(o) \wedge C \sqsubseteq D\}$
and $M_b(o, C_b) = \mu_{\sqsupseteq}\{D \in O_b \mid O_b \vDash_{OWL} D \sqsubseteq C_b \wedge \neg D(o)$ *and* $\nexists C'_a \in O_a; A_{ab} \vDash \langle C'_a, \geq, D\rangle\}$
with $\mu_r S = \{C \in S \mid C' \in S \wedge O \vDash_{OWL} C' r C \Rightarrow C = C'\}$.

This corresponds to the operators as illustrated in Figure 6.4 (p.73).

7.6.3 Results

With respect to the properties introduced before, it can be proved [VAN DEN BERG, ATENCIA, and EUZENAT 2021], that:

- all presented operators are correct;
- all operators are incomplete (the logical model is far stronger than ARG agents);
- delete and addjoin are (unilaterally) partially redundant, i.e. for some agents, the upgrade does not provide more information than the announcement.

There are two ways to interpret these results:

- Adaptive (ARG) agents are logically weak, and
- The logical model is not precise enough as it does not closely model ARG agents.

This is the result of a trade-off between adaptive and logical agents. The logic could be made closer to the adaptive agents, or logical agents may be implemented. More work is necessary for better understanding the mismatch between these two worlds.

Conclusion

In conclusion, dynamic epistemic logics can model cooperative cognitive agents. However, they make strong assumptions on the capabilities of agents and rely on sophisticated semantic structures. They remain a good guide for modelling the communication of distributed knowledge.

They also show some limitations:

- These are mostly propositional logics;
- They rely on models built from above (the theories contain all knowledge and belief of agents);
- We did not took advantage of nested formulas, such as $K_a B_b(K_c\phi \vee B_d\psi)$;
- We only presented public announcements, but in agent communication private (group) announcements would be more adapted.

Quiz

- In modal logic, is $S \vDash S'$ defined by $\forall M, M \vDash S \Rightarrow M \vDash S'$ or by $\forall M, \forall w \in W_M, M, w \vDash S \Rightarrow M, w \vDash S'$? Does one expression implies the other? Why?
- What is the difference between the modalities for knowledge (K) and belief (B), informally, semantically, axiomatically?
- In multi-agent epistemic logics, how does the semantics make knowledge and beliefs differ?
- In the presented epistemic logic, how is the relation between knowledge and beliefs dealt with?
- How is belief revision performed in the presented dynamic epistemic logic?

References

[ZACK 2020] Richard Zack, Boxes and diamonds: an open introduction to modal logic, 2020 https://bd.openlogicproject.org/.

[FAGIN, HALPERN, MOSES, and VARDI 1995] Ronald Fagin, Joseph Halpern, Yoran Moses, Moshe Vardi, Reasoning about knowledge, The MIT press, Cambridge (MA US), 1995

[VAN DITMARSCH, VAN DER HOEK, and KOOI 2018] Hans van Ditmarsch, Wiebe van der Hoek, Barteld Kool, Dynamic epistemic logic, Springer, Heidelberg (DE), 2018

8

Conclusions

These notes are in no way conclusive: this course is here to open perspectives to students about the numerous ways to approach distributed knowledge and its semantics. Each of these chapters are far from exhausting their topic. On the contrary, it is expected that they opened the reader's appetite for digging more deeply.

I tried to consider, at a simple level, different approaches to knowledge heterogeneity and dynamics that can be witnessed every day. I focussed on the sheer semantic problems that they raise. I attempted to connect such approaches as much as possible. However a lot of work remains to be done to theoretically and practically connect them better.

How it will develop next, is up to the stimulated reader.

Quiz

- What is meant by 'semantics'?
- For what defining the semantics of a logic or language is useful? Or, what can be done with the semantics that cannot be done without it?
- Is this a problem if two agents express their knowledge (or beliefs) differently? Why?
- What does it mean for a structure to be inconsistent?
- How may errors in the interpretation of terms or formulas be identified?
- What is a model in this course?
- What does it means that a particular structure (formula, theory, etc.) entails another?
- What does it mean for a structure to be inconsistent?
- Is this a problem that two agents' knowledge do not have the same models? Why?
- What does it means that two sets of knowledge have no common model?
- What are the two possible interpretations of the word 'knowledge' in this course?
- What is the benefit of connecting several knowledge sources together?

A

The model theoretic pattern, in a nutshell

During this course, I will use the model-theoretic semantic pattern which consists of:

- A *language* $\mathcal{L}$ for expressing *statements* $\phi \in \mathcal{L}$ and sets of statements $S \subseteq \mathcal{L}$ (graph, ontology, alignment);
- Its semantics defined through *interpretations*: how an interpretation I satisfies a statements $s \in \mathcal{L}$, $I \vDash \phi$;
- A *model* of S is an interpretation satisfying all statements in S; the set of models of S is $\mathcal{M}(S)$).
- A statement ϕ is a *consequence* of a set of statements S ($S \vDash \phi$) if it satisfies all models of S; the set of consequences of S is noted $Cn(S) = \{\phi \mid S \vDash \phi\}$.
- A set of statements with no model is said *inconsistent*; otherwise *satisfiable*.
- A set of statements S is *entailed* by, or *consequence* of, another S' ($S' \vDash S$) if all models of S' are models of S;
- A *SELECT query* $q[\overrightarrow{X}]$ is a statement of $\mathcal{L}$ in which some entities are replaced by *variables* in $\overrightarrow{X}$; the *answers* $\mathcal{A}(q[\overrightarrow{X}], S)$ to the query $q[\overrightarrow{X}]$ in S, is the set of *variable assignments* σ such that $S \vDash \sigma(q[\overrightarrow{X}])$;
- A deductive system (or a reasoner) is able, from a set of axioms S, to find/check whatever statements (*completeness*) and only those (*correctness*) it entails ($Cn(S)$).

  These notions entertain some covariance relations:

$$S \subseteq S' \Rightarrow \mathcal{M}(S) \supseteq \mathcal{M}(S') \Rightarrow \begin{array}{c} S' \vDash S \\ Cn(S) \subseteq Cn(S') \end{array}$$

The $Cn : 2^{\mathcal{L}} \rightarrow 2^{\mathcal{L}}$ function is a closure operation. This means that it satisfies the three properties:

(extensivity)	$S \subseteq Cn(S)$
(monotonicity)	$S \subseteq S' \Rightarrow Cn(S) \subseteq Cn(S')$
(idempotency)	$Cn(S) = Cn(Cn(S))$

These are articulated as in Figure A.1.

Why does it matter? Because everything that will be implemented in a computer is in the position of a deductive system whose correctness, and completeness, will be judged with respect to the defined semantics. In this course, I only discuss the semantics. But it is desirable that implemented systems follow it and not the other way around.

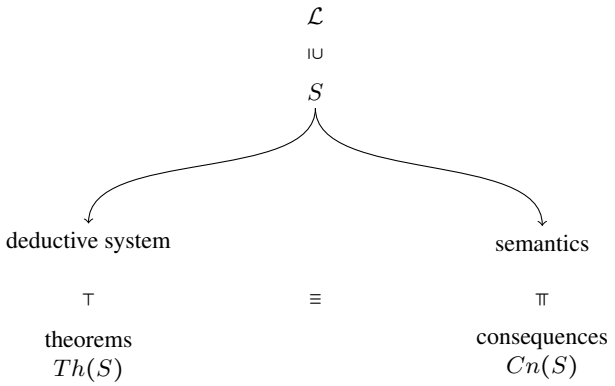


Figure A.1. The basic scheme of a (computational) logic.

B

Graphic notation

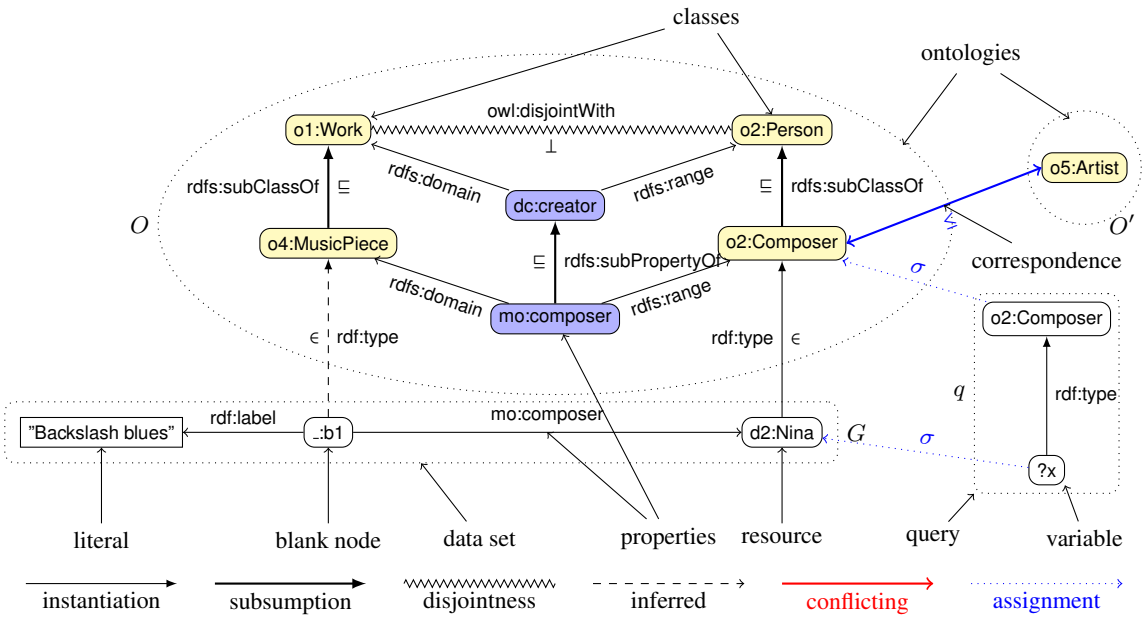


Figure B.1. Notation used in ontology graphs.

C

Assignments

With this course I am used to provide assignments between the sessions. Students are expected to send me their results by email before the next session (since lectures occur in the morning, it is better to do this the morning before). About the assignments:

- They are not exercises that should require long thinking.
- I assume that they should be allocated around 15 minutes.
- Assignments build on top of each other, but students can change the data/ontology/alignments of their previous answer if this make things more meaningful.
- The main point of the exercise is that students try and identify problems and that the lecture can address questions that they have already asked themselves.
- They are briefly and anonymously discussed at the beginning of each course to clarify misunderstandings and missing points.
- This should encourage students to think about the course, and to consider it globally.

Assignment #1

The first assignment is to write down between five and ten short sentences (I mean short) in English about things you know about a specific topic. Here we take the topic of 'Grenoble'. There is not more constraint than that.

C.1 Assignment #2

Assignment #2, is the following:

- Express some (or all) of the sentences you provided for the first assignment in RDF;
- Provide an ontology defining the terms and properties that you use.

Assignment #3

Provide an alignment between your ontology of Assignment #2 (or another ontology corresponding to the assignement) and the ontology made of the concepts provided below. What this alignment allows to deduce (entails), that was not entailed without the alignment?

The piece of ontology O_1 of Listing C.1 is to be considered:

```
my:locatedIn          rdfs:domain          my:Place .
my:locatedIn          rdfs:range           my:Place .
my:InhabitedPlace     rdfs:subClassOf      my:Place .
my:nbInhabitant       rdfs:domain          my:InhabitedPlace .
my:nbInhabitant       rdfs:range           xsd:positiveInteger .
my:nameInhabitant     rdfs:domain          my:Settlement .
my:nameInhabitant     rdfs:range           xsd:string .
```

```
my:Settlement            rdfs:subClassOf          my:InhabitedPlace .
my:Municipality          rdfs:subClassOf          my:Settlement .
my:Building              rdfs:subClassOf          my:InhabitedPlace .
my:Building              owl:disjointWith         my:Municipality .
my:Building              rdfs:subClassOf          _:b0 .
 _:b0                    owl:onProperty           my:locatedIn .
 _:b0                    owl:onClass              my:Municipality .
 _:b0                    owl:qualifiedCardinality 1 .

my:Event                 rdfs:subClassOf          _:b1 .
 _:b1                    owl:onProperty           my:date .
 _:b1                    owl:onClass              xsd:date .
 _:b1                    owl:qualifiedCardinality 1 .
my:location              rdfs:domain              my:Event .
my:location              rdfs:range               my:Place .
my:nbInSeries            rdfs:domain              my:Event .
my:nbInSeries            rdfs:range               xsd:positiveInteger .
my:label                 rdfs:domain              my:Event .
my:label                 rdfs:range               xsd:string .

my:Event                 owl:disjointWith         my:Place .
```

Listing C.1. Piece of ontology O_1 about places and events.

as well as foaf:Person and the whole foaf ontology (you can look at it at http://xmlns.com/foaf/spec/).

Assignment #4

Translate the three conjunctive queries of Listing C.2 using your alignments:

```
// Find the cities where there is a monument called 'La bastille'
SELECT ?city
WHERE {
    ?X          rdf:type        my:Building .
    ?X          rdfs:label      "La bastille" .
    ?X          my:locatedIn    ?city .
    ?city       rdf:type        my:Municipality .
}

// List all events and their city of occurence
SELECT ?event ?city
WHERE {
    ?event      rdf:type        my:Event .
    ?event      my:location     ?city .
    ?city       rdf:type        my:Settlement  .
}

// Who is the mayor of Grenoble.
SELECT ?firstname ?lastname
WHERE {
    ?X          rdf:type        my:Municipality .
    ?X          rdfs:label      "Grenoble" .
    ?X          my:hasMayor     ?Y .
    ?Y          foaf:givenName  ?firstname .
    ?Y          foaf:familyName ?lastname .
}
```

Listing C.2. Three queries q_1, q_2 and q_3 against ontology O_1.

Do you think that these query would provide the expected answers and why?

Assignment #5

Consider a new ontology (O_2 of Listing C.3):

```
o2:House          rdfs:subClassOf o2:Place .
o2:JailHouse      rdfs:subClassOf o2:House .
o2:OperaHouse     rdfs:subClassOf o2:House .
o2:CityHall       rdfs:subClassOf o2:House .
o2:Fortress       rdfs:subClassOf o2:House .
o2:AdminArea      rdfs:subClassOf o2:Place .
```

```
o2:Town          rdfs:subClassOf o2:AdminArea .
o2:GeoArea       rdfs:subClassOf o2:Place .
o2:Mountain      rdfs:subClassOf o2:GeoArea .
o2:River         rdfs:subClassOf o2:GeoArea .
o2:Hill          rdfs:subClassOf o2:GeoArea .
o2:Lake          rdfs:subClassOf o2:GeoArea .

o2:House         owl:disjointWith o2:AdminArea .
o2:House         owl:disjointWith o2:GeoArea .

d:p0             rdf:type         o2:Place .
d:p0             rdfs:label       "France" .

d:p1             rdf:type         o2:Town .
d:p1             rdfs:label       "Grenoble" .
d:p1             my:locatedIn     d:p0 .

d:p2             rdf:type         o2:Town .
d:p2             rdfs:label       "Paris" .
d:p2             my:locatedIn     d:p0 .
d:p1             owl:differentFrom d:p2 .

d:p3             rdf:type         o2:Hill .
d:p3             my:locatedIn     d:p1 .
d:p3             rdfs:label       "Bastille" .
d:p3             rdf:type         o2:JailHouse .
d:p3             rdf:type         o2:OperaHouse .
d:p3             my:locatedIn     d:p2 .
d:p3             rdfs:label       "Bastille" .
```

Listing C.3. Ontology and graph O_2 about the Bastille.

with its alignment $A_{2,1}$ to the ontology O_1 provided in Assignment #3:

```
o2:Place      <    my:Place
o2:House      <    my:Building
o2:AdminArea  >    my:Municipality
o2:Town       <    my:Municipality
```

Listing C.4. Alignment $A_{2,1}$ between O_2 and O_1.

- Is ontology O_2 consistent? Coherent?
- Is the alignment $A_{2,1}$ consistent? Coherent?
- Is the network of ontologies $\langle\{O_1, O_2\}\{A_{2,1}\}\rangle$ consistent? Coherent?
- If no, can you identify statements that cause the inconsistency or incoherence? (can you tell what is wrong?)
- Is there, assuming that one of these statements were just added, local ways to restore consistency?
- What is the best solution to solve the problem in your opinion? Is it a partial meet revision?

Assignment #6

Define a protocol for evolving your ontologies:

- Identify and describe what could be an interaction using your current ontology and data (possibly alignment).
- Define what could be a failure in this interaction.
- Describe which adaptation could be applied to the knowledge (ontology, alignment) to improve knowledge in reaction to the failure.

That may lead you to modify your ontology or your alignment.

The main point of this is that you think a bit about it; that you try.

Assignment #7

Provide a logical model of what agents know (ontologies) and believe (alignments) about Grenoble.

Explain how announcing some of their beliefs will lead them to evolve their models. In particular, consider agents which may consider that 'La bastille' is a place in Grenoble or in Paris and the effect of public announcements and conservative upgrades that 'La bastille' is a place in Grenoble.

References

[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 (cited p. 52).

[ADJIMAN et al. 2006] Philippe ADJIMAN, Philippe CHATALIC, François GOASDOUÉ, Marie-Christine ROUSSET, and Laurent SIMON, Distributed Reasoning in a Peer-to-Peer Setting: Application to the Semantic Web, *Journal of artificial intelligence research* 25:269–314, 2006 (cited p. 52).

[ALCHOURRÓN et al. 1985] Carlos ALCHOURRÓN, Peter GÄRDENFORS, and David MAKINSON, On the logic of theory change: partial meet contraction and revision functions, *Journal of symbolic logic* 50(2):510–530, 1985 (cited pp. 54, 55, 57).

[ALKHATEEB et al. 2009] Faisal ALKHATEEB, Jean-François BAGET, and Jérôme EUZENAT, Extending SPARQL with regular expression patterns (for querying RDF), *Journal of web semantics* 7(2):57–73, 2009 (cited p. 41).

[ANTONIOU and VAN HARMELEN 2008] Grigoris ANTONIOU and Frank VAN HARMELEN, *A semantic web primer*, 2nd ed., The MIT press, Cambridge (MA US), 2008 (cited p. 25).

[AUSTIN 1962] John AUSTIN, *How to do things with words*, Harvard university press, Cambridge (MA US), 1962 (cited pp. 67, 84).

[AXELROD 1997] Robert AXELROD, The dissemination of culture: a model with local convergence and global polarization, *Journal of conflict resolution* 41(2):203–226, 1997 (cited p. 68).

[BAADER et al. 2003] Franz BAADER, Diego CALVANESE, Deborah MCGUINNESS, Daniele NARDI, and Peter Patel SCHNEIDER, eds., *The description logic handbook: theory, implementations and applications*, Cambridge University Press, 2003 (cited p. 17).

[BAGET 2005] Jean-François BAGET, RDF Entailment as a Graph Homomorphism, in: *Proc. 4th International Semantic Web Conference (ISWC), Galway (IE)*, pp. 82–96, 2005 (cited p. 12).

[BALTAG and SMETS 2008] Alexandru BALTAG and Sonja SMETS, A qualitative theory of dynamic interactive belief revision, in: Giacomo BONANNO, Wiebe VAN DER HOEK, and Michael WOOLDRIDGE, eds., *Logic and the Foundations of Game and Decision Theory*, vol. 3, Amsterdam university press, Amsterdam (NL), pp. 22–58, 2008 (cited p. 80).

[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 (cited p. 17).

[BECKETT 2009] Dave BECKETT, *OWL 2 Web Ontology Language Document Overview*, Recommendation, W3C, 2009 (cited pp. 17, 25).

[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 (cited p. 30).

[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 (cited pp. 10, 13).

[CARROLL and KLYNE 2004] Jeremy CARROLL and Graham KLYNE, *RDF concepts and abstract syntax*, Recommendation, W3C, 2004 (cited p. 8).

[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 (cited p. 30).

[Dalal 1988] Mukesh Dalal, Investigations into a theory of knowledge base revision: preliminary report, in: *Proc. 7th national conference on artificial intelligence (AAAI), Philadelphia (PA US)*, pp. 475–479, 1988 (cited p. 58).

[David et al. 2011] Jérôme David, Jérôme Euzenat, François Scharffe, and Cássia Trojahn, The Alignment API 4.0, *Semantic Web Journal* 2(1):3–10, 2011 (cited p. 37).

[Dean and Schreiber 2004] Mike Dean and Guus Schreiber, *OWL Web Ontology Language: reference*, Recommendation, W3C, 2004 (cited p. 17).

[Euzenat 2007] Jérôme Euzenat, *Semantic web semantics*, en, Lecture notes, université Joseph Fourier, Grenoble (FR), 190 pp., 2007 (cited pp. 8, 10, 13, 21, 28, 30, 39, 44).

[Euzenat 2014a] — First experiments in cultural alignment repair (extended version), in: *Proc. ESWC 2014 satellite events revised selected papers*, Lecture notes in computer science 8798, pp. 115–130, 2014 (cited p. 70).

[Euzenat 2014b] — *The category of networks of ontologies*, Research report 8652, INRIA, 2014 (cited pp. 62, 64).

[Euzenat 2015] — Revision in networks of ontologies, *Artificial intelligence* 228:195–216, 2015 (cited pp. 34, 37, 59, 63, 65, 66).

[Euzenat 2017] — Interaction-based ontology alignment repair with expansion and relaxation, in: *Proc. 26th International Joint Conference on Artificial Intelligence (IJCAI), Melbourne (VIC AU)*, pp. 185–191, 2017 (cited pp. 70, 72, 75).

[Euzenat and Shvaiko 2013] Jérôme Euzenat and Pavel Shvaiko, *Ontology matching*, 2nd ed., Springer, Heidelberg (DE), 2013 (cited pp. 28, 37, 49).

[Fagin et al. 1995] Ronald Fagin, Joseph Halpern, Yoram Moses, and Moshe Vardi, *Reasoning about knowledge*, The MIT press, Cambridge (MA US), 1995 (cited pp. 35, 83, 88).

[Falakh et al. 2023] Faiq Miftakhul Falakh, Sebastian Rudolph, and Kai Sauerwald, *AGM Belief revision, semantically*, tech. rep. 2112.13557, arXiv, 2023 (cited pp. 54, 58, 66).

[Fermé and Hansson 2018] Eduardo Fermé and Sven Ove Hansson, *Belief change: introduction and overview*, Springer, Cham (CH), 2018 (cited pp. 55, 66).

[Flouris et al. 2006] Giorgos Flouris, Dimitris Plexousakis, and Grigoris Antoniou, On Generalizing the AGM Postulates, in: *Proc. 3rd European Starting AI Researcher Symposium (STAIRS)*, pp. 132–143, 2006 (cited p. 57).

[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 (cited p. 31).

[Gärdenfors 1992] Peter Gärdenfors, ed., *Belief revision*, Cambridge university press, Cambridge (UK), 1992 (cited p. 55).

[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 (cited pp. 30, 31).

[Glimm and Ogbuji 2013] Birte Glimm and Chimezie Ogbuji, *SPARQL 1.1 entailment regimes*, Recommendation, http://www.w3.org/TR/sparql11-entailment, W3C, 2013 (cited pp. 39, 44).

[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 (cited pp. 16, 42).

[Hansson 1996] Sven Ove Hansson, A test battery for rational database updating, *Artificial intelligence* 82:341–352, 1996 (cited p. 55).

[Harris and Seaborne 2013] Steve Harris and Andy Seaborne, *SPARQL 1.1 query language*, Recommendation, http://www.w3.org/TR/sparql11-query, W3C, 2013 (cited pp. 39, 47, 52).

[Hayes 2004] Patrick Hayes, *RDF Semantics*, Recommendation, W3C, 2004 (cited pp. 8, 10, 12, 15, 16).

[Hayes and Patel-Schneider 2014] Patrick Hayes and Peter Patel-Schneider, *RDF Semantics*, Recommendation, http://www.w3.org/TR/rdf11-mt/, W3C, 2014 (cited pp. 10, 16).

[HITZLER et al. 2009] Pascal HITZLER, Markus KRÖTZSCH, and Sebastian RUDOLPH, *Foundations of semantic web technologies*, Chapman & Hall/CRC, 2009 (cited p. 25).

[HORROCKS et al. 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 (cited p. 17).

[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 (cited pp. 9, 12).

[KATSUNO and MENDELZON 1991a] Hirofumi KATSUNO and Alberto MENDELZON, On the difference between updating a knowledge base and revising it, in: *Proc. 2nd international conference on principles of knowledge representation and reasoning (KR), Cambridge (MA US)*, pp. 387–394, 1991 (cited p. 56).

[KATSUNO and MENDELZON 1991b] — Propositional knowledge base revision and minimal change, *Artificial intelligence* 52(3):263–294, 1991 (cited pp. 54, 56–58).

[KIRBY et al. 2008] Simon KIRBY, Hannah CORNISH, and Kenny SMITH, Cumulative cultural evolution in the laboratory: An experimental approach to the origins of structure in human language, *Proceedings of the National Academy of Sciences* 105(31):10681–10686, 2008 (cited p. 68).

[LENZERINI 2002] Maurizio LENZERINI, Data Integration: A Theoretical Perspective, in: *Proc. 21st Symposium on Principles of Database Systems (PODS)*, pp. 233–246, 2002 (cited p. 30).

[MESOUDI 2011] Alex MESOUDI, *Cultural Evolution: How Darwinian theory can explain human culture and synthesize the social sciences*, University of Chicago Press, Chicago (IL US), 2011 (cited pp. 67, 68, 75).

[MOTIK 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 (cited p. 21).

[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 (cited p. 14).

[NEBEL 1994] Bernhard NEBEL, Base revision operations and schemes: semantics, representation, and complexity, in: *Proc. 11th European Conference on Artificial Intelligence (ECAI)*, pp. 341–345, 1994 (cited pp. 56, 58).

[PATEL-SCHNEIDER et al. 2004] Peter PATEL-SCHNEIDER, Patrick HAYES, and Ian HORROCKS, *OWL Web ontology language semantics and abstract syntax*, Recommendation, W3C, 2004 (cited p. 21).

[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 (cited pp. 39, 41, 42).

[PLOTKIN 1993] Henry PLOTKIN, *Darwin machines and the nature of knowledge*, Harvard university press, Cambridge (MA US), 1993 (cited p. 67).

[POLLERES 2007] Axel POLLERES, From SPARQL to rules (and back), in: *Proc. 16th World Wide Web Conference (WWW)*, pp. 787–796, 2007 (cited p. 39).

[PRUD'HOMMEAUX and SEABORNE 2008] Eric PRUD'HOMMEAUX and Andy SEABORNE, *SPARQL Query Language for RDF*, Recommendation, W3C, 2008 (cited pp. 28, 39).

[SCHREIBER and RAYMOND 2014] Guus SCHREIBER and Yves RAYMOND, *RDF Primer*, Recommendation, https://www.w3.org/TR/rdf11-primer/, W3C, 2014 (cited p. 25).

[STEELS 2012] Luc STEELS, ed., *Experiments in cultural language evolution*, John Benjamins, Amsterdam (NL), 2012 (cited pp. 68–70, 75).

[VAN DEN BERG et al. 2021] Line VAN DEN BERG, Manuel ATENCIA, and Jérôme EUZENAT, A logical model for the ontology alignment repair game, *Autonomous Agents and Multi-Agent Systems* 35(2):1–34, 2021 (cited pp. 85, 87, 88).

[VAN DITMARSCH et al. 2018] Hans VAN DITMARSCH, Wiebe VAN DER HOEK, and Barteld KOOI, *Dynamic Epistemic Logic*, Springer, Heidelberg (DE), 2018 (cited p. 88).

[ZACK 2020] Richard ZACK, *Boxes and diamonds: an open introduction to modal logic*, Open logic project, 2020 (cited p. 88).

[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 (cited p. 30).

[ZIMMERMANN 2013] — Logical formalisms for agreement technologies, in: Sasha OSSOWSKI, ed., *Agreement technologies*, Springer Verlag, Heidelberg (DE), chap. 5, pp. 69–82, 2013 (cited p. 30).

[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 (cited p. 30).

Index

Semantics of distributed knowledge

Jérôme Euzenat

Knowledge is widely distributed within our society. It has to be accessed and combined to be properly exploited.

These notes address the problem in dealing with distributed knowledge. They first consider how to represent knowledge and especially classifications or ontologies. Because there may be different such representations, they are aligned in order to create networks of ontologies. These networks enable answering queries based on connected distributed knowledge and data. But their heterogeneity may lead knowledge to be inconsistent or contradictory. Belief revision is then used in order to restore consistency within networks of ontologies. However, other ways to deal with distributed knowledge and its evolution exist. One of these, cultural knowledge evolution, consists of selecting knowledge in an evolutionary way. It will be considered experimentally, through multi-agent simulations, and theoretically through attempting at modelling it with dynamic epistemic logics.

The mentioned topics are not explored in depth. Instead, the tight relationships between these are considered from a semantic standpoint.