Lecture Notes
Ontology Engineering

Department of Computer Science
University of Cape Town, South Africa

Version 1.2.1
2015

C. Maria Keet

Lecture Notes
Ontology Engineering

C. Maria Keet

Keywords: Ontology, ontologies, Knowledge base, Description Logics, OWL, Semantic Web, Ontology Development

These Lecture Notes are typeset in LaTeX

Cover design by Maria Keet
Printed and bound by you

Contents

Preface

These lecture notes for the Ontology Engineering course (at the University of Cape Town, South Africa) are structured in the order of teaching the module for a 4th year BSc(honours) in Computer Science. This 10-credit course covers Block I and Block II; the third block (advanced topics) will not be covered, and has not been updated in v1.2 and v1.2.1 of the lecture notes.

Given that the field is in flux, much new material is becoming available rapidly, so these notes will be updated regularly. I have written these notes under time constraints; hence, a note of caution: *these lecture notes are a work in progress* and feedback is welcome. A bright side of it being an active field of research and technology development is that you will become acquainted with the forefront of computing—instead of the more common relatively 'established' knowledge of your undergraduate courses—which will put you at a competitive advantage and give you a head-start especially if you plan to find a job after graduating with the BSc(hons). The, perhaps downside, is that, at the time of writing, there is no text book yet that fills the gap between undergraduate and PhD-student & researcher-level handbooks and scientific articles in logic-based ontologies, so you will have make do with these notes, the slides, lectures that provide further explanation, exercises with answers, and you will have to learn to read several of those scientific articles.

The contents of these lecture notes are themselves updated versions of material I developed for the EMCL Semantic Web Technologies course I taught at the Free University of Bozen-Bolzano, Italy, in 2009,subsequent syllabi for the Ontology Engineering courses at the University of Havana and University of Computer Science, Cuba, and at the Masters Ontology Winter School 2010, South Africa, and the COMP718/720 notes of 2012 and 2013 at UKZN, and the OE honours course at UCT. Some contents of these lecture notes, or accompanying slides used in the lectures, or associated exercises are adapted from slides or tutorials made by other people, and I would like to thank them for having made that material available for use and reuse. They are (in alphabetic order) Jos de Bruijn, Diego Calvanese, Nicola Guarino, Matthew Horridge, Ian Horrocks, Markus Krötzsch, Lina Lubyte, Tommie Meyer, Mariano Rodríguez-Muro, Sebastian Rudolph, František Simančík, and David Toman.

Cape Town, South Africa
January, 2015.

C. Maria Keet

Course Outline

0.1 Aims and Synopsis

The principal aim of this module is to provide the participant with a comprehensive overview of ontology engineering. A secondary aim is to provide hands-on experience in ontology development and Semantic Web Technologies that illustrate the theory, such as language features, automated reasoning, and top-down and bottom-up ontology development.

This module covers material such that, upon completion, the student:

(i) has a general understanding of the notion of what ontologies and knowledge bases are, what they can be used for, how, and when not;

(ii) has obtained an understanding of the, currently, main ontology languages—OWL and its underlying Description Logics languages—in order to represent the knowledge in ontologies formally and to reason over them, and have a basic understanding of what the automated reasoner does;

(iii) can confidently use an Ontology Development Environment;

(iv) can confidently use methods and methodologies to develop ontologies, including the top-down approach with foundational ontologies and bottom-up using non-ontological resources such as relational databases, natural language or thesauri; and

(v) has become acquainted with several major applications and application scenarios, such as the Semantic Web, and has had a taste of the research trends in the field.

Interwoven in the module's aims is skills development for the students' BSc(honours) project. The students will become familiar with reading scientific literature and will gain experience in report writing and presenting their work to their peers.

0.2 Module assessment

The final mark for the module will be based on three categories of assessment:

- A test (exam) at the end of the course [50%]
- One intermediate practical assignment [15%]

  The assignment has to be handed in by the end of week x (exact date to be communicated when the lecture schedule is set); details of the assignment will be communicated separately.
- Mini-project due at the end of the course [30%]

The topics you can choose from will be communicated in the first week of commencement of the lectures; you must have chosen a topic in the second week (earlier is better, and first-come-first-serve) (exact date to be communicated when the lecture schedule is set)
- Contribution to the OE Semantic Wiki [5%]

Note: Something has to be submitted for the test (exam), practical assignment, and mini-project in order to have a chance to pass the course. Contributing to the OE Semantic Wiki is not compulsory but highly recommended.

0.3 Course material

The course material consists of:
- These lecture notes;
- Lecture slides posted on the course's Vula site (the slides are a lecture aid, not a summary of the contents);
- Reading material for the lectures: the lecture notes (relevant chapter/section to be read before the lecture), and the papers, book chapters, standardization documentation that are listed at the end of each chapter.
- Material for the chosen mini-project (some are already listed in the bibliography, and additional ones or will be made available depending on the chosen project topic).

The slides and reading material will be made available through the course's Vula site. Some material in the slides and what will pass the revue during the lecture has not been integrated in these lecture notes yet (and not everything described in the lecture notes appear in the slides), so one is not a substitute for the other.

0.4 Course content

Note: each "lecture" is a double lecture, and this is a rough indication of the schedule.

1. *Lecture 1: Introduction.* The introductory lecture addresses differences between databases and knowledge bases, conceptual data models and ontologies, what an ontology is (and is not), and a prominent application area, being the Semantic Web.

2. **Block 1: Logic foundations for ontologies**

   (a) *Lecture 2: FOL recap and tableaux.* The first part of the lecture will be a recap of the basics of first order predicate logic, introduce the notion of model-theoretic semantics, and how to formalise knowledge based on natural language and diagrams. The second part introduces reasoning over a logical theory, and tableaux in particular.

   (b) *Lecture 3: Description Logics and OWL.* This lecture is devoted to the basics of Description Logics (DLs, a family of languages that are decidable fragments of FOL and lie at the basis of most 'species' of the World Wide Web consortium's standardised Web Ontology Language OWL), and we will start introducing OWL 2 features.

   (c) *Lecture 4: OWL 2 and Automated Reasoning.* We continue with OWL 2, and its profiles will be discussed. In addition, we take a look at the principal automated reasoning services for (OWL) ontologies, such as satisfiability checking and classification.

3. **Block 2: Ontology engineering**

   (a) *Lecture 5: Methods and Methodologies.* This lecture takes a closer look at parameters for ontology design, methods (such as OntoClean, glassbox reasoning) and tools, and more comprehensive methodologies (Methontology, MoKi, NeOn methodology).

   (b) *Lecture 6: Top-down Ontology Development I.* One step of ontology development is the use of foundational ontologies and their formalisations (on paper in FOL, in OWL DL, Isabelle). In particular, we shall look at the DOLCE and BFO foundational ontologies, and commence with a core relation in ontology development, being part-whole relations.

   (c) *Lecture 7: Top-down Ontology Development II.* We continue with part-whole relations and add the notions ontology design patterns and ontology reuse.

   (d) *Lecture 8: Bottom-up Ontology Development.* In addition to starting from 'above', one can reusing legacy material to generate candidate classes and relations to speed up populating an ontology. In particular, we will look at relational databases, thesauri (including SKOS), and natural language processing.

4. **Block 3: Advanced Topics**

   Note: we will not cover this material, but is made available for your perusal.

   (a) *Lectures x and y: 'Ontology'/Conceptual Model-based Data Access.* Due to various usage scenarios, there is a need to maintain the link between the data and the knowledge, such as in scientific workflows or *in silico* biology and enhanced user and content management in e-learning applications. This can be done in one knowledge base or to connect a database to an ontology (or vv.) so that one can query the database 'intelligently' through the ontology by availing of its classes, object properties and axioms. In these two lectures, we will start with a motivation and an overview of one such system (WONDER), which relies on the DL-Lite family of DL languages (roughly OWL 2 QL), a mapping layer, and a relational database. We then shall look at its technical details as well as the principal options for reaslising such a system.

   (b) *Lecture z: Temporal ontologies.* There are various extensions to the 'basic' ontology languages and reasoning services, such as vagueness, uncertainty, and the temporal dimension. In this lecture we cover a temporal DL and some of the modelling issues it solves.

5. **Block 4: Recap and Miniprojects**

   (a) *Lecture 9: Overflow, Recap, and Research trends.* Time permitting (or on request), we take a look as some research trends, where we will take a selection from any of the following topics: modularization, dealing with imprecise knowledge (fuzzy/rough), interaction with conceptual data modelling, multilingual ontologies, or 'debugging' ontologies (glass-box reasoning).

(b) *Mini-project Presentations.* Pending scheduling, each group will present the outcome of their chosen mini-project and discuss it in class, and everyone must attend the lecture and participate in the presentation and discussions.

CHAPTER 1

Introduction

This chapter introduces ontologies: what they are (roughly), what they are used for, and describes a few success stories where they have been instrumental at solving problems. Where and how an ontology can solve problems is not of the variety "when you have only a hammer, everything looks like a nail", but where the use of an ontology was *the* solution to a particular problem, or at least an essential ingredient. A warning upfront: some paragraphs may sound a bit 'esoteric', and lots of new terms are introduced in this chapter that are fleshed out in much more detail only in subsequent chapters. Therefore, it is probably useful to revisit this chapter later—and don't be put off if it is not all clear and raises many questions now!

A very short and informal way of clarifying what "an ontology" in computing is, is that it is a file containing structured knowledge about a particular subject domain, and this file is used as a component of a so-called 'intelligent' information system. Fancy marketing talk may speak of some of those **ontology-driven information systems** as "like a database *on steroids*!". Ontologies have been, and are being, used to solve data integration problems by providing the *common, agreed-upon vocabulary* which is then used in a way so that the software understands that, say, an entity Student of a relational database DB_1 actually means the same thing as AdvancedLearners in some application software OO_2. Tools can then be developed to link up those two applications and exchange information. Over time, people invented other ways to use ontologies and contribute to solving different problems. For instance, a question-answering system that lets the scientist chat with a library chatterbot to find relevant literature in agriculture (compared to imprecise string and rigid keyword matching), automatically find a few theoretically feasible candidate rubber molecules out of very many (compared to painstaking trial-and-error work in the laboratory), and automated discovery of a new enzyme (outperforming the human experts!).

In the next section (Section 1.1), we have a quick peek at what an ontology—the artefact—looks like, and proceed to the more and less pedantic viewpoints of defining what an ontology is (Section 1.2). Sections 1.3 and 1.4 provide a flavour of the usefulness and some success stories of ontologies.

1.1 What does an ontology look like?

The actual artefact can appear in many formats that are tailored to the intended 'user', but at the heart of it, is a logic-based representation. Let us take as example the African Wildlife Ontology (AWO), which is a so-called 'tutorial ontology' that will return in the exercises. The AWO contains knowledge about wildlife, such as that giraffes eat leaves and twigs, that they are herbivores, that herbivores are animals, and so on.

A mathematician may prefer to represent such knowledge with first order predicate logic. For instance:

$$\forall x(Lion(x) \rightarrow \forall y(eats(x,y) \land Herbivore(y)) \land \exists z(eats(x,z) \land Impala)) \tag{1.1}$$

that states that "all lions eat herbivores, and they also eat impalas". Thus, this axiom could be one of the axioms in the ontology. One can represent the same knowledge also in logics other than some plain vanilla first order logic; e.g., in a **Description Logic** language, we have the same knowledge formally represented as:

$$\mathsf{Lion} \sqsubseteq \forall \mathsf{eats.Herbivore} \sqcap \exists \mathsf{eats.Impala} \tag{1.2}$$

A domain expert, however, typically will prefer a more user-friendly rendering, such as an automatically generated (pseudo) natural language rendering, e.g.:

Each lion eats only herbivore and eats some Impala

where the first "∀' in equation 1.1 is verbalised as Each, the "∧" as and, and the "∃" as some. Pseudo-natural language is also called 'controlled natural language' because it uses only a specific subset of a complete natural language. Another option is to use a graphical language, as shown in Figure 1.1.

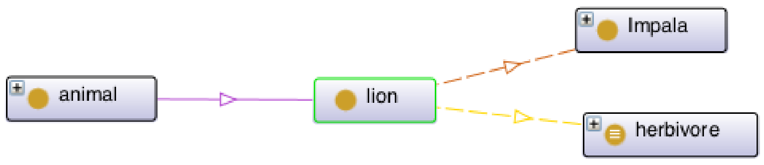


Figure 1.1: Screenshot of the lion eating only herbivores and at least some impala (with the OntoGraf plugin in Protégé 4.x).

Remember that an ontology is an engineering artefact that has to have a machine-processable format that faithfully adheres to the logic. None of these representations are easily computer-processable, however. To this end, there are **serialisations** of the ontology that are easily computer-processable; the most popular one is the **Web Ontology language OWL**, which actually consists of several languages and has several machine-processable formats. Some of those serialisations are still for human consumption (sort of), others are definitely not designed with human readability in mind, but really for computers and the tools that use ontologies. A machine-processable version of the class lion in the RDF/XML format looks as follows:

```
<owl:Class rdf:about="&AfricanWildlifeOntology1;lion">
   <rdfs:subClassOf rdf:resource="&AfricanWildlifeOntology1;animal"/>
   <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="&AfricanWildlifeOntology1;eats"/>
        <owl:someValuesFrom rdf:resource="&ontologies;AfricanWildlifeOntology2.owl#Impala"/>
      </owl:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
```

```
      <owl:Restriction>
        <owl:onProperty rdf:resource="&AfricanWildlifeOntology1;eats"/>
        <owl:allValuesFrom rdf:resource="&AfricanWildlifeOntology1;herbivore"/>
      </owl:Restriction>
   </rdfs:subClassOf>
   <rdfs:comment>Lions are animals that eat only herbivores.</rdfs:comment>
</owl:Class>
```

where the "∀" from equation 1.1 is serialised as `owl:allValuesFrom`, the "∃" is serialised as `owl:someValuesFrom`, and the subclassing ("→" and "⊑") as `rdfs:subClassOf`. You typically will not have to write an ontology in RDF/XML format. As a computer scientist, you may design tools that will have to process or modify such machine-processable ontology files (though even there, there are tool development toolkits and APIs that cover many tasks). For the design of an ontology, there are ontology development environments that render the ontology graphically, textually, or with an logic view[1]. A screenshot of one such tool, Protégé, is included in Figure 1.2. We will use mainly Protégé in the course.

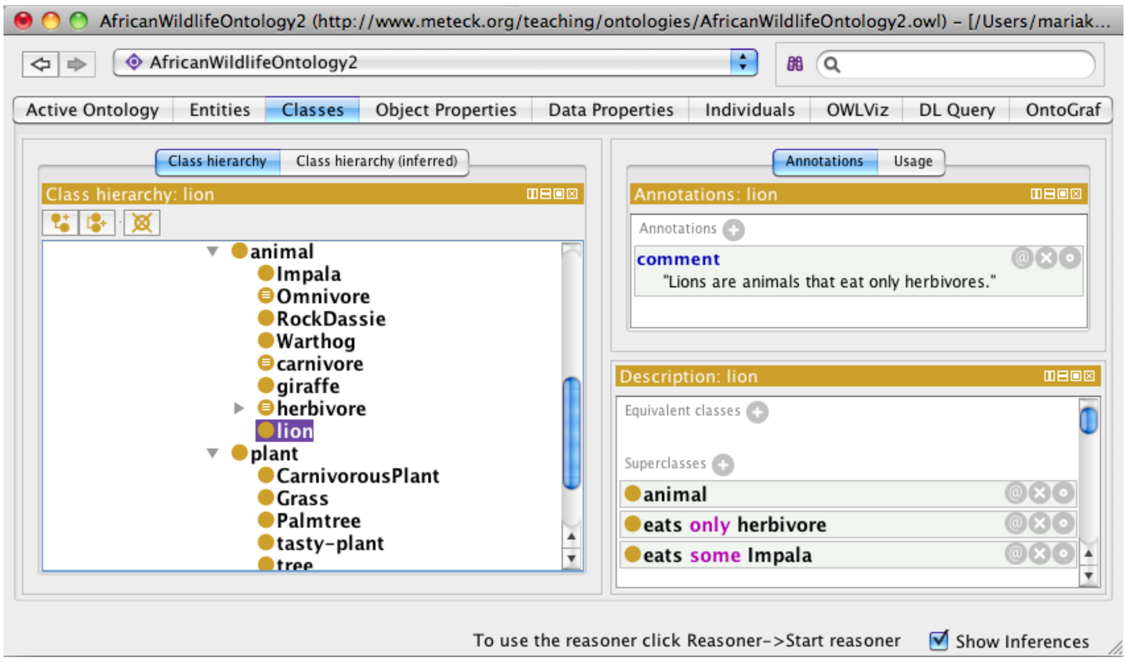


Figure 1.2: Screenshot of the lion eating only herbivores and at least some impala in the Protégé ontology development environment.

1.2 What is an ontology?

Note: You may prefer to read this section again later on in the course, when we are well into Block II. Try to read it now anyway, but if it's not clear upon the first read, then don't worry, as it will become clearer as we go along.

To place "ontologies" in its right context, the first two questions one has to ask and answer are:

- What is an ontology?

[1]ODE tools have lots of other features, which you will use in the exercises

- What is it good for? (or: what problems does it solve?)

In order to arrive at the answers, let us first compare it with some artefacts you are already familiar with: relational databases and conceptual data models such as EER and UML.

It is sometimes stated that "knowledge bases are more powerful relational databases", but that does not explain what the 'umph' is that offer the additional features. A down to earth comparison between relational databases and **knowledge bases** reveals that, unlike RDBMSs, knowledge bases include the *representation of the knowledge explicitly*, by having rules included, by using *automated reasoning* (beyond plain queries) *to infer new or implicit knowledge and detect inconsistencies* of the knowledge base, and they usually operate under the Open World Assumption[2].

An important distinction between conceptual data models and ontologies is that a conceptual data model provides an application-specific implementation-independent representation of the data that will be handled by the prospective application, whereas (domain) ontologies provide an *application-independent representation* of a specific subject domain (in principle, regardless the particular application). From this distinction follow further differences regarding their contents—in theory at least—to which we shall return to later in the module. Looking at actual ontologies and conceptual data models, the former is normally formalised in a logic language, whereas conceptual modelling is more about drawing the boxes and lines informally[3], and they are used differently and serve different purposes.

This informal brief comparison gives a vague idea of what an ontology might be, but it does not get us closer to a definition of what an ontology is. There is no unanimously agreed-upon definition what an ontology is and definitions that have been proposed within computer science have changed over the past 20 years (this will be discussed during the lecture). The most quoted (but problematic!) definition is the following one by Tom Gruber:

Definition 1.1 (by [Gruber, 1993])**.** *An ontology is a specification of a conceptualization.*

You may see this quote especially in older scientific literature on ontologies, but it has been superseded by other, more precise ones. Gruber's definition is unsatisfactory for several reasons: what is a "conceptualization" exactly, and what is a "specification"? A proposed refinement to address these two questions is the following one:

Definition 1.2 (by [Studer et al., 1998])**.** *An ontology is a formal, explicit specification of a shared conceptualization.*

However, this still leaves us with the questions as to what a "conceptualization" is and what a "formal, explicit specification" is. And why—and how—"shared"? A comprehensive definition is given in Guarino's landmark paper on ontologies [Guarino, 1998] (revisited in [Guarino et al., 2009]):

Definition 1.3 (by [Guarino, 1998])**.** *An ontology is a logical theory accounting for the* intended meaning *of a formal vocabulary, i.e. its* ontological commitment *to a particular* conceptualization *of the world. The intended models of a logical language using such a vocabulary are constrained by its ontological commitment. An ontology indirectly reflects this commitment (and the underlying conceptualization) by approximating these intended models.*

[2]vs. Closed World Assumption in a relational database setting. We return to the OWA and CWA later

[3]though one surely can provide formal foundations for conceptual data models (e.g., [Artale et al., 2007a, Berardi et al., 2005, Keet, 2013])

To some, this a mouthful and too wieldy, and a simpler definition is given by the developers of the World Wide Web Consortium's standardised ontology language OWL:

Definition 1.4 (by [Horrocks et al., 2003]). *an ontology being equivalent to a Description Logic knowledge base*[4].

That last definition has a different issue, and is unduly restrictive, because 1) it surely is possible to have an ontology that is represented in another logic language (OBO format, Common Logic, etc.) and 2) then formalising a thesaurus as a "Description Logic knowledge base" (i.e., in OWL) also ends up as a simple 'light-weight ontology' (e.g., the NCI thesaurus[5] as cancer 'ontology'[6]) and a conceptual data model in EER or UML that is translated into OWL becomes an 'application ontology' or 'operational ontology' by virtue of it being formalised in OWL. But, as we saw above, there are differences between the two.

An alternative approach to the issue of definitions was taken in the 2007 Ontolog Communiqué[7], where its participants and authors made a collection of things drawn into diagram to express 'things that have to do with an ontology'; this is depicted in Figure 1.3. It is intended as a "Template for discourse" about ontologies, which has a brief[8] and longer[9] explanation of the text in the labeled ovals.

A broader scope of the ontological level is also described in [Guarino, 2009], and a more recent overview about definitions of "an ontology" versus Ontology in philosophy can be found in [Guarino et al., 2009], which refines in a step-wise and more precise fashion Studer et al's [Studer et al., 1998] and Guarino's [Guarino, 1998] definition (Definitions 1.2 and 1.3, above).

For better or worse, currently, and in the context of the most prominent application area of ontologies—the **Semantic Web**—the tendency is toward it being equivalent to a logical theory, and a Description Logics knowledge base in particular (Definition 1.4). Ontologist at least frown when someone calls 'a thesaurus in OWL' or 'an ER diagram in OWL' ontologies, but even aside from that: the blurring of the distinctions between the different artefacts is problematic for various reasons, and one should note the fact that just because something is represented in OWL does not make it an ontology, just as that something that is represented in a language other than OWL may well be an ontology.

Orthogonally, there are discussions about what is actually represented in an ontology, i.e., its contents, from a philosophical perspective. Philosophers are in the picture because the term 'ontology' is taken from philosophy, where it has a millennia-old history, and one uses insights emanating from philosophy when developing good ontologies. When we refer to that philosophical notion, we use **Ontology**, with a capital 'o', and it does not have a plural. One debate is about ontology as a representation of a conceptualization—roughly: things you are thinking of—and as a representation of reality. Practically, whether that is a relevant topic depends on the subject domain for which you would be developing an ontology. If you represent formally the knowledge about, say, malaria infections, you would better represent the (best approximation of) reality, being the current state of scientific knowledge, not some divergent political or religious opinion about it, because the wrong representation can lead to wrong inferences, and therewith wrong treatments that are either ineffective or even harmful. Conversely, there are subject domains where it does not

[4]Note: we will go into some detail of OWL and Description Logics in Chapters 3 and 4.

[5]`http://ncit.nci.nih.gov/`

[6]`http://www.mindswap.org/2003/CancerOntology/nciOncology.owl`

[7]`http://ontolog.cim3.net/cgi-bin/wiki.pl?OntologySummit2007_Communique`

[8]`http://ontolog.cim3.net/cgi-bin/wiki.pl?OntologySummit2007_FrameworksForConsideration/DimensionsMap`

[9]`http://ontolog.cim3.net/cgi-bin/wiki.pl?OntologySummit2007_Communique`

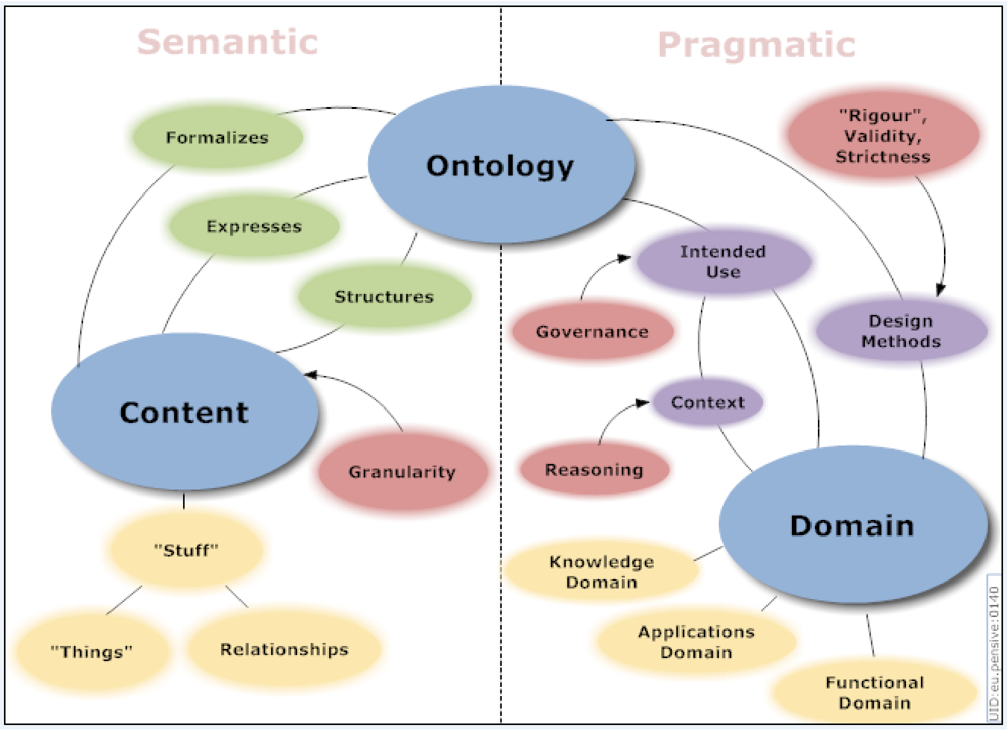


Figure 1.3: The OntologySummit2007's "Dimension map".

really matter much whether you represent reality or a conceptualization thereof, or something independent of whether that exists in reality or not, or even certainly does not exist in reality. We will discuss some of the pros and cons of these views in the lecture; a recent, longer, debate can be found in [Merrill, 2010a, Merrill, 2010b, Smith and Ceusters, 2010]. Merrill [Merrill, 2010a] provides several useful clarifications. First, there is an "*Empiricist Doctrine*" where "the terms of science... are to be taken to refer to actually existing entities in the real world", such as Jacaranda tree, HIV infection and so forth, which are considered mind-independent, because (roughly) HIV infections occurred also without humans thinking of it, knowing how it worked, and naming those events HIV infections. This is in contrast with the "conceptualist view according to which such terms refer to concepts (which are taken to be psychological or abstract formal entities of one sort or another)", with concepts considered to be mind-dependent entities; prototypical examples of such mind-dependent entities are Phlogiston and Unicorn—there are no objects in the world as we know it that are phlogiston or unicorns, only our outdate theories and fairy tale stories, respectively, about them. Second, the "*Universalist Doctrine*", which asserts "that the so-called "general terms" of science" (HIV infection etc.) "are to be understood as *referring directly to universals*", with universals being "a class of mind independent entities, usually contrasted with individuals, postulated to ground and explain relations of qualitative identity and resemblance among individuals. Individuals are said to be similar in virtue of sharing universals." [MacLeod and Rubenstein, 2005]. However, philosophers do not agree on the point whether universals exist, and even if they exist, what kind of things they are. This brings the inquiring person to metaphysics, which, perhaps, is not necessarily crucial in building ontologies that are to serve information systems; e.g., it need not be relevant for developing an ontology about viruses whilst adhering to the empiricist doctrine. The philosophically inclined reader may wish to go a step further and read about interactions between Ontology and metaphysics

[Varzi, 2012].

1.3 What is the usefulness of an ontology?

Ontologies for information systems were first proposed to contribute to solving the issues with **data integration**: an ontology provides the common vocabulary for the applications that is at one level of abstraction higher up than conceptual data models such as EER diagrams and UML Class Diagrams. Over the years, it has been used also for other purposes. We start with two distinct scenarios of data integration where ontologies play a central role, and subsequently describe other scenarios where ontologies are an important part of the solution.

1.3.1 Data and information system integration

Figure 1.4 sketches the idea of the ontology-driven schema-based and conceptual data model-based data integration and Figure 1.6 shows an example of *data-level data integration* that we shall elaborate on in the next two subsections.

Integrating legacy systems. In the setting of ontology-driven schema-based and conceptual data model-based data integration, a typical situation is as follows. You have several databases in the same topic; e.g., this was the case when UDW and UND merged into UKZN: each university had its own database with information about students, yet, as UKZN, there had to be one single database to manage the data of all students. This meant that the two databases had to be integrated somehow. A similar situation occurs oftentimes in industry, especially due to mergers and acquisitions, in government due to the drive for e-Government services to the citizens of the country, or attempting to develop a software system for integrated service delivery, as well as in healthcare due to a drive for electronic health records that need to combine various systems, say, a laboratory database with the doctor's database, among other scenarios.

While the topic of data integration deserves its own course, we focus here only on the ontology-driven aspect. Let us assume we have the relational databases and therewith at least their respective physical schemas, and possibly also the relational model and even the respective conceptual models, and also some object-oriented application software on top of the relational database. Their corresponding conceptual data models are tailored to the RDBMS/OO application and may or may not be modelled in the same conceptual data modelling language; e.g., one could be in EER, another on ORM, in MADS, in UML and so forth. The example in Figure 1.4 is a sketch of such a situation about information systems of flower shops, where at the bottom of the figure we have two databases and one application that has been coded in C++. In the layer above that, there is a section of their respective conceptual data models: we have one in EER with bubble-notation, one in ORM, and one UML Class Diagram. Each conceptual data model has "Flower" and "Colour" included in some way: in the UML Class diagram, the colour is an attribute of the flower, i.e., $\mathsf{Color} \mapsto \mathsf{Flower} \times \mathtt{String}$ (that actually uses only the values of the Pantone System) and similarly in the EER diagram (but then without the data type), and in the ORM diagram the colour is a value type (unary predicate) Colour with an additional relation to the associated datatype `colour region` in the spectrum. Clearly, the notion of the flower and its colour is the same throughout, even though it is represented differently in the conceptual data model and in the implementations. It is here that the ontology comes into play, for it is *the* place to assert exactly that underlying, agreed-upon notion. It enables one to assert that:

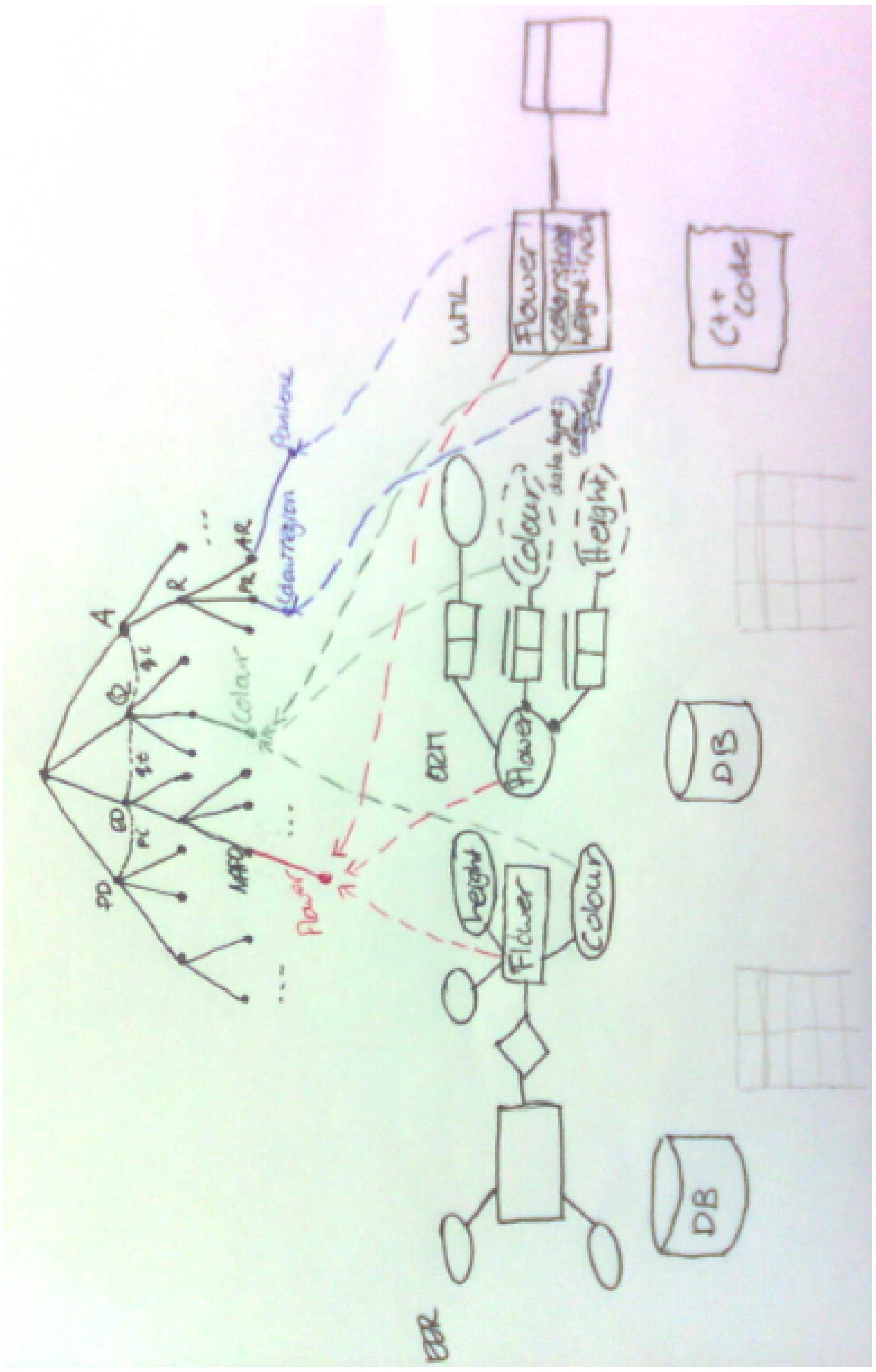


Figure 1.4: Sketch of an ontology-based application integration scenario. Bottom: different implementations, such as relational databases and OO software; Centre: conceptual data models tailored to the application; Top: a domain ontology that provides a shared common vocabulary for interoperability among the applications. See text for explanation.

- EER's, ORM's and UML diagram's Flower 'means' *Flower* in the domain ontology[10], which is indicated with the red dashed arrows.
- EER's and ORM's Colour and UML's Color denote the same kind of thing, albeit at one time it is represented as a unary predicate (in ORM) and other times it is a binary relation with a data type, i.e., an attribute. Their 'mappings' to the entity in the ontology (green dashed arrows), *Colour*, indicates that agreement.
- There is no agreement among the conceptual models when it comes to the data type used in the application, yet they may be mapped into their respective notion in an ontology (blue dashed arrows). For instance, the *ColourRegion* for the values of the colour(s) in the colour spectrum is a *PhysicalRegion*, and one might say that the *PantoneSystem* of colour encoding is an *AbstractRegion*.

The figure does not include names of relationships in the conceptual data model, but they obviously can be named at will; e.g., hasColour in the ORM diagram. Either way, there is, from an ontological perspective, a specific type of relation between the class and its attribute: one of dependency or inherence (roughly: that specific colour instance depends on the existence of the flower: if that particular flower does not exist, then that specific instance of it does not exist). An ontology can provide those generic relations, too. In the sketch, this happens to be the *qt* relationship between enduring objects (like flowers) and the qualities they have (like their colour), and from the quality to the value regions (more precisely: qualia), the relation is called *ql* in Figure 1.4.

Although having established such links does not complete the data integration, it is the crucial step—the rest has become, by now, largely an engineering exercise.

Generation of conceptual models for related new systems (preventing interoperability problems). Instead of a data integration scenario from existing applications, one also can start with an ontology, and take it from there to develop various conceptual data models for specific applications [El-Ghalayini et al., 2006, Jarrar et al., 2003, Sugumaran and Storey, 2006], a bit like the Enterprise Models you may have come across in information system design. In this way, interoperability is guaranteed upfront because of the shared elements in the new conceptual data models. For instance, a parthood relation between objects—as is depicted in Figure 1.5—or a participation relation between an object and a process (e.g., a surfer *participates in* a surfing event) is reused in the conceptual data model. Either way, the ontology provides a shared common vocabulary for interoperability among the applications. The orchestration of the principal components will be described in more detail during the lecture, given that this is still an important usage area. We will see more about contents of the ontologies, including foundational ontologies, later on in the course.

Data-level data integration. While in computer science the aforementioned two approaches to data integration was under investigation, domain experts in molecular biology needed a quick and practical solution to the data integration problem urgently. Having noticed the idea of ontologies, they came up with another approach, being interoperability at the instance-level, tuple-by-tuple, or even cell-by-cell, and that with multiple databases over the Internet instead of the typical scenario of RDBMSs within an organisation. This is done with lightweight ontologies, or **structured controlled vocabularies**. The basic idea is illustrated in Figure 1.6. There are multiple databases, which in the figure are the KEGG and InterPro databases. In the KEGG database,

[10]that in this case is linked to a *foundational ontology*, DOLCE, and there it is a subclass of a Non-Agentive Physical Object; we return to this in Block II

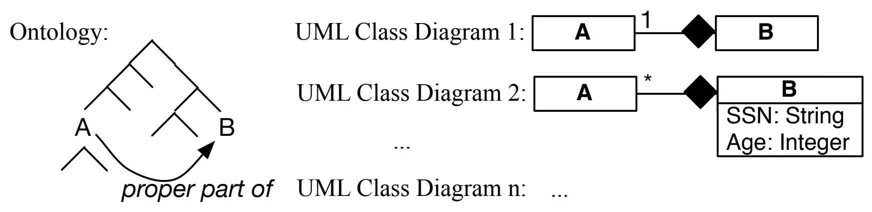


Figure 1.5: Basic illustration of taking information from an ontology and using it for several conceptual data models (here: UML Class Diagrams), where the constraints may be refined or attributes added, yet sharing the semantics of A, B, proper part of, and that A is a proper part of B.

there is a tuple with as key K01834 and it has several attributes (columns in the table in the relational database), such as the name (`gpmA`), and further down in the display there is an attribute `Other DBs`, which has as entry `GO:0004619`; i.e., there is a tuple in the table along the line of ⟨K01834, . . . , GO:0004619⟩. In the InterPro database, we have a similar story but then for the entity with the key IPR005995, where there is a section "GO Term Annotation" with an attribute `function` that has `GO:0004619`; i.e., a tuple ⟨IPR005995, . . . , GO:0004619⟩. That is, the two clearly distinct tuples—each with their separate identifier from a different identifier scheme, with different attributes, one physically stored in a database in Japan and the other in the USA—actually talk about the same thing: GO:0004619, or: Phosphoglycerate Mutase Activity.

The "GO:0004619" is an identifier for a third artefact: a class in the Gene Ontology (GO) [Gene Ontology Consortium, 2000]. The GO is a structured controlled vocabulary that contains the classes and their relationship that the domain experts agree upon (over 20000 entities by now). The curators of the two databases each annotated their entity with a GO term, and thereby they assert they have to do with that same thing, and therewith have created an entity-level linking and interoperability through the GO. Practically, on top of that, these fields are hyperlinked (in the soft copy: blue text in KEGG and green underlined text in the screenshot of the InterPro entry), so that a vast network of data-level interlinked databases has been created. Also the GO term is hyperlinked to the GO file online, and in this way, you can browse from database to database availing of the terms in the ontology without actually realising they are wholly different databases, but instead appear like one vast network of knowledge.

There are many more such ontologies, and several thousand databases that are connected in this way, not only thanks to ontologies, but where the ontology serves as the essential ingredient to the data integration. Some scientific journals require the authors even to use those terms from the ontologies when they write about their discoveries, so that one more easily can find papers about the same entity[11].

1.3.2 Ontologies as part of a solution to other problems

Over the years, ontologies have been shown to be useful in a myriad of other application scenarios; among others, negotiation between software services, mediation between software agents, bringing more quality criteria into conceptual data modelling to develop a better model (hence, a better quality software system), orchestrating the components in

[11]It used to be a sport among geneticists to come up with cool names for the genes they discovered (e.g., "Sonic hedgehog"); when a gene was independently recovered, each research team typically had given the gene a different name, which can end up as a Tower of Babel of its own that hampers progress in science.

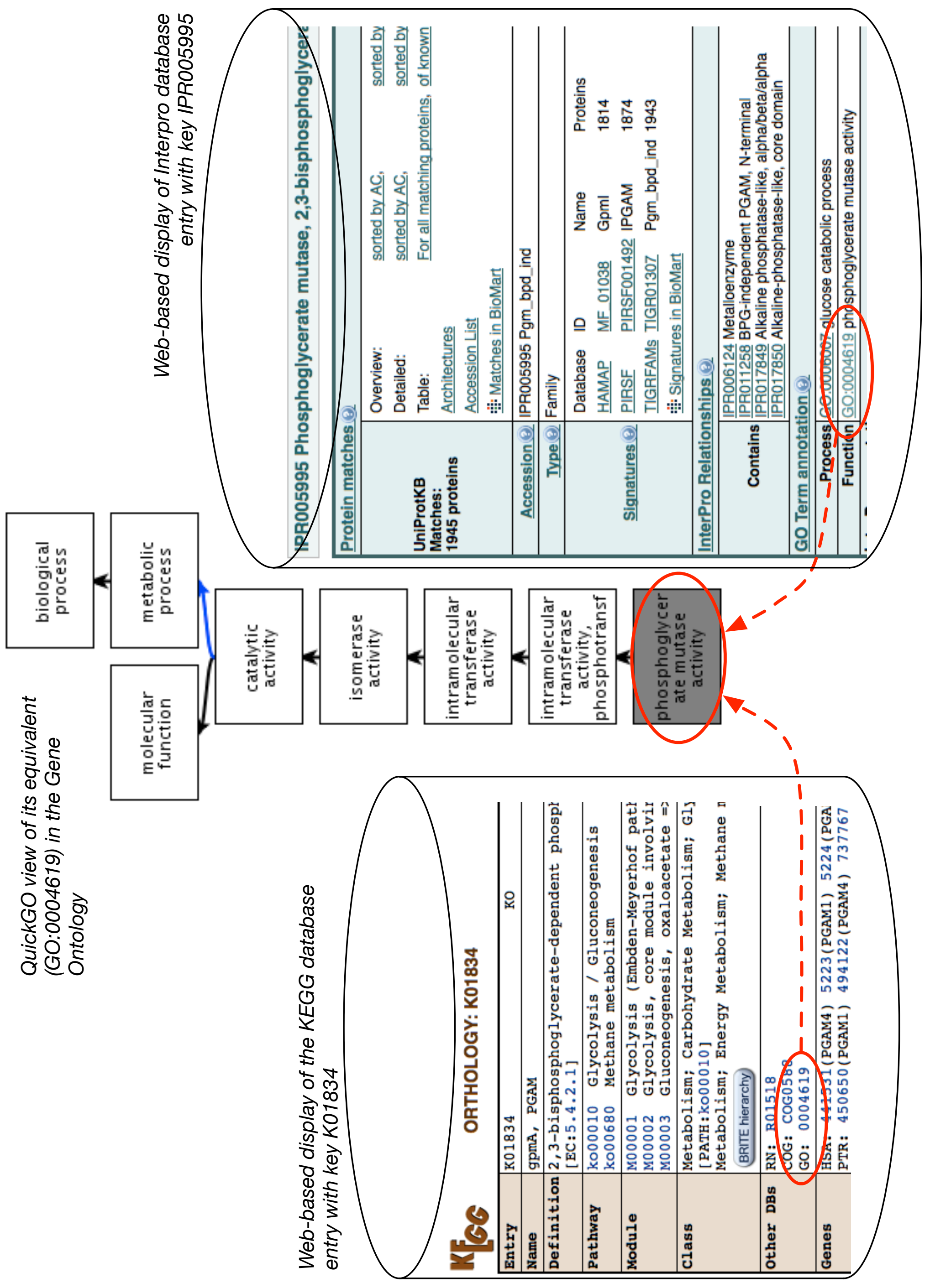

Web-based display of Interpro database entry with key IPR005995
IPR005995 Phosphoglycerate mutase, 2,3-bisphosphoglycerate
Protein matches
UniProtKB Matches: 1945 proteins
Accession IPR005995 Pgm_bpd_ind
Type Family
Signatures
InterPro Relationships
Contains
GO Term annotation
Process GO:0006007 glucose catabolic process
Function GO:0004619 phosphoglycerate mutase activity
QuickGO view of its equivalent (GO:0004619) in the Gene Ontology
biological process
metabolic process
molecular function
catalytic activity
isomerase activity
intramolecular transferase activity
intramolecular transferase activity, phosphotransf
phosphoglycerate mutase activity
Web-based display of the KEGG database entry with key K01834
KEGG
ORTHOLOGY: K01834
Entry K01834 KO
Name gpmA, PGAM
Definition 2,3-bisphosphoglycerate-dependent phosph [EC:5.4.2.1]
Other DBs RN: R01518 COG: COG0588 GO: 0004619

Figure 1.6: Illustration of ontology-based data-level integration: two databases, the KEGG and InterPro, with a web-based front-end, and each database has its data (each tuple in the database, where possible) annotated with a term from the Gene Ontology.

semantic scientific workflows, e-learning, ontology-based data access, top-k information retrieval, management of digital libraries, improving the accuracy of question answering systems, and annotation and analysis of electronic health records. Some of these applications will pass the revue later in the module and three are briefly illustrated in this section. Note, however, that this does not mean that ontologies are the panacea for everything, and some ontologies are better suitable to solve one or some of the problems, but not others. Put differently, it is prudent to keep one's approach to engineering: conduct a problem analysis first, collect the requirements and goals, and then assess if an ontology indeed is part of the solution or not. If it is part of the solution, then we enter in the area of **ontology engineering**.

e-Learning

The 'old-fashioned' way of e-learning is a so-called content-push: the lecturer sends out softcopies of the notes, slides, answers to the solutions, and perhaps the video recordings of the lectures, and the student consumes it. This is a one-size-fits-all approach regardless the student's background with acquired knowledge and skills, and learning preferences and habits, which cannot be assumed to be homogeneous in an e-learning setting, or at least much less so than with respect to your fellow students in the ontology engineering class. A more sophisticated way for e-learning is *adaptive e-learning*, which tailors the contents to the student based on prior knowledge and learning habits. To be able to automatically tailor the offering to the student, one has to develop a 'smart' e-learning application that can figure out what kind of student is enrolled. Put differently: students have certain properties (part-time/full-time student, age, undergraduate degree, etc.), the learning objects have to be annotated (by skill level and topic), and user logs have to be categorised according to type of learning pattern, and based on that the material and presentation can be adjusted, like skipping the section on first order logic and delve deeper into, or spend more time on, ontology engineering and modelling if you have a mathematics background, whereas a philosopher may crave for more content about foundational ontologies but skip reverse engineering of relational databases, or offer a student more exercises on a topic s/he had difficulties with. This requires **knowledge representation**—of the study material, questions, answers—and **automated reasoning** to classify usage pattern and student, and annotated content, i.e., using ontologies and knowledge bases to make it work solidly and in a repeatable way. See, e.g., [Henze et al., 2004] as a start for more details on this topic.

Deep question answering

Watson[12] is a sophisticated question answering engine that finds answers to trivia/general knowledge questions for the *Jeopardy!* TV quiz that, in the end, did consistently outperform the human experts of the game. For instance, a question could be "who is the president of South Africa?": we need algorithms to parse the question, such as that 'who' indicates the answer has to be a person and a named entity, it needs to be capable to detect that South Africa is a country and what a country is, and so on, and then have some kind of a look-up service in knowledge bases and/or natural language documents to somehow find the answer by relating 'president', 'South Africa' and 'Jacob Zuma' and that he is the current president of the country. An ontology can then be used in the algorithms of both the understanding of the question and finding the right answer[13] and

[12]`http://en.wikipedia.org/wiki/Watson_(computer)`

[13]On a much more modest scale, as well as easier accessible and shorter, Vila and Ferrández describe this principle and demonstrated benefits for their Spanish language based question-asnwering system in

integrating data sources and knowledge, alongside natural language processing, statistical analysis and so on, with the 'and so on' comprising more than 100 different techniques[14]. Thus, a key aspect of the system's development was that one cannot go in a linear fashion from natural language to knowledge management, but have to use an integration of various technologies, including ontologies, to make a successful tool.

Semantic Scientific Workflows

Due to the increase in a variety of equipment, their speed, and decreasing price, scientists are generating more data than ever, and are collaborating more. This data has to be analysed and managed. In the early days, and, to some extent, to this day, many one-off little tools were developed, or simply scripted together with PERL or Ruby on Rails, used once or a few times and then left for what it was. This greatly hampers repeatability of experiments, insight in the provenance of the data, and does not quite follow a methodological approach for so-called *in silico* biology research. Over the past 10 years, comprehensive software and hardware infrastructures have been, and are being, built to fix these and related problems. Those IT 'workbenches' (as opposed to the physical ones in the labs) are realised in *semantic scientific workflow systems*. An example of the virtual bench is Taverna [Goble et al., 2007]. This, in turn can be extended further; e.g., to incorporate data mining in the workflow, as depicted in Figure 1.7. This contains ontologies of both the subject domain where it is used as well as ontologies about data mining itself that serve to find appropriate models, algorithms, datasets, and tools for the task at hand, depicted in Figure 1.8. Thus, we have a large software system—the virtual workbench—to facilitate scientists to do their work, and some of the components are ontologies for integration and data analysis across the pipeline.

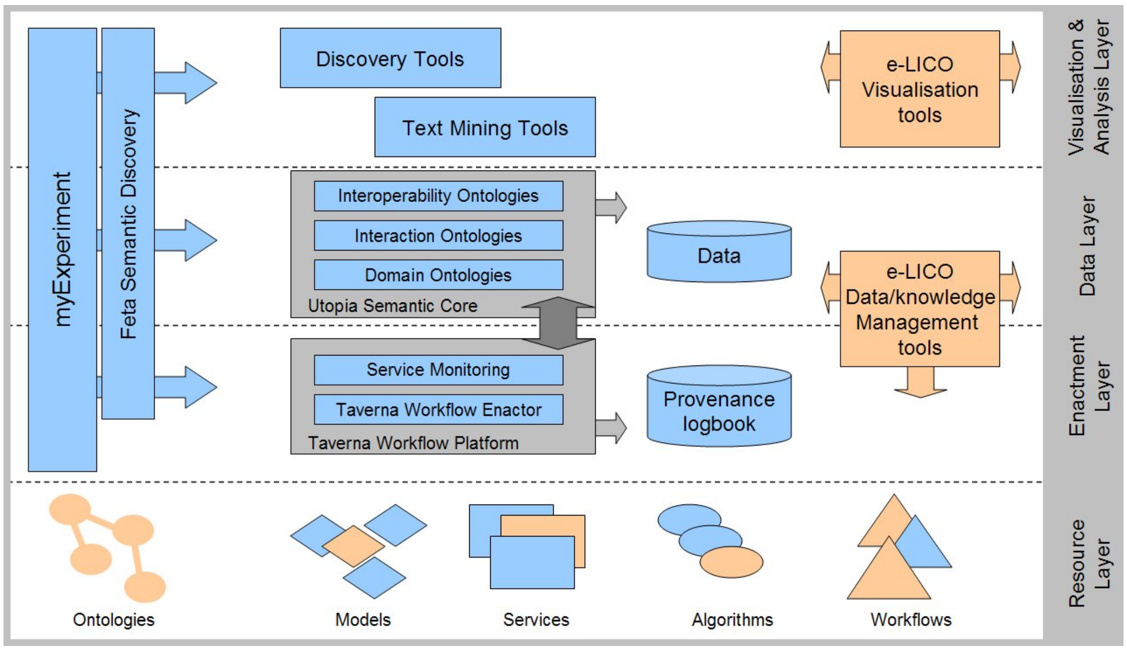


Figure 1.7: Overview of the architecture of the e-Laboratory for Interdisciplinary Collaborative Data Mining (Source: `http://www.e-lico.eu/?q=node/17`).

the agricultural domain [Vila and Ferrández, 2009].

[14]`ftp://public.dhe.ibm.com/common/ssi/ecm/en/pow03061usen/POW03061USEN.PDF`

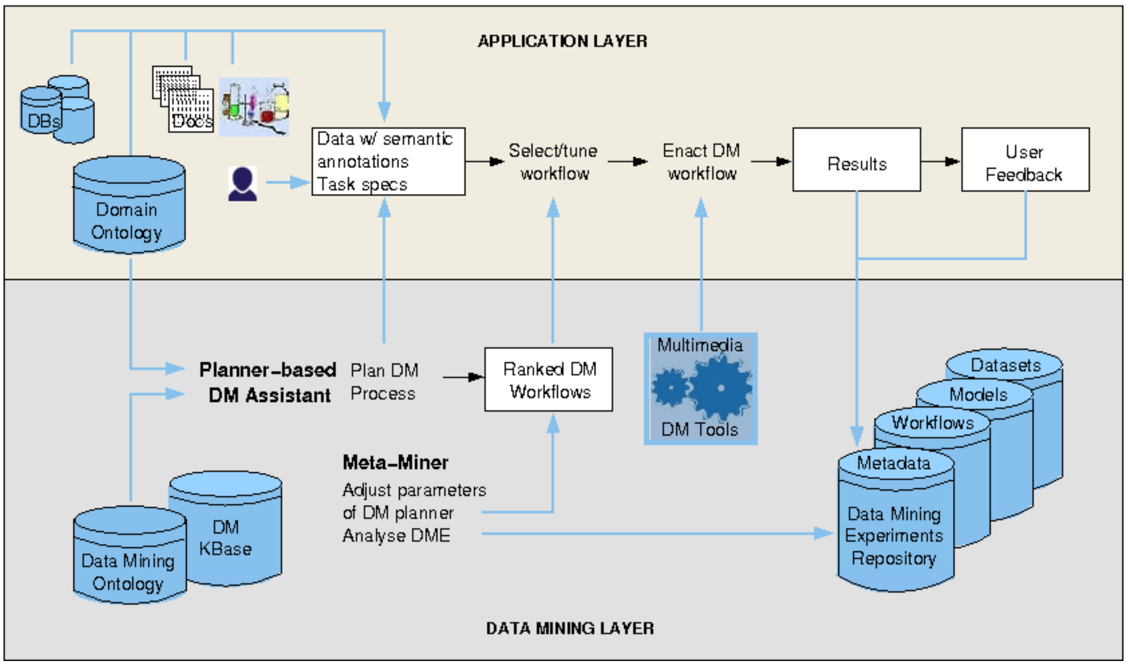


Figure 1.8: The data mining and application layers of the e-Laboratory for Interdisciplinary Collaborative Data Mining (Source: `http://www.e-lico.eu/?q=node/17`).

1.4 Success stories

To be able to talk about successes of ontologies, and its incarnation with Semantic Web Technologies in particular, one first needs to establish when something can be deemed a success, when it is a challenge, and when it is an outright failure. Such measures can be devised in an absolute sense—compare technology x with an ontology-mediated one: does it outperform on measure y?—and relative—to whom is technology x deemed successful? During the lecture, we will illustrate some of the successes, whereas challenges are deferred to later on in the course.

A major success story of the development and use of ontologies for data linking and integration is the Gene Ontology [Gene Ontology Consortium, 2000], its offspring, and subsequent coordinated evolution of ontologies [Smith et al., 2007] within the OBO Foundry Project[15]. These frontrunners from the Gene Ontology Consortium and their colleagues in bioinformatics were adopters of some of the Semantic Web ideas even before Berners-Lee, Hendler, and Lassila wrote their Scientific American paper in 2001 [Berners-Lee et al., 2001], even though they did not formulate their needs and intentions in the same terminology: they did want to have shared, controlled vocabularies with the same syntax to facilitate data integration—or at least interoperability—across Web-accessible databases, have a common space for identifiers, it needing to be a dynamic, changing system, to organize and query incomplete biological knowledge, and, albeit not stated explicitly, it all still needed to be highly scalable [Gene Ontology Consortium, 2000]. That is, bioinformaticians and domain experts in genomics already organized themselves together in the Gene Ontology Consortium[16], which was set up officially in 1998, to realize a solution for these requirements. The results exceeded anyone's expectations in its success for a range of reasons. Many tools for the Gene Ontology (GO) and its common Knowledge Representation format, `.obo`, have been developed, and other research

[15]`http://www.obofoundry.org/`
[16]`http://www.geneontology.org/`

groups adopted the approach to develop controlled vocabularies either by extending the GO, e.g., rice traits, or adding their own subject domain, such as zebrafish anatomy and mouse developmental stages. This proliferation, as well as the OWL development and standardization process that was going on at about the same time, pushed the goal posts further: new expectations were put on the GO and its siblings and on their tools, and the proliferation had become a bit too wieldy to keep a good overview what was going on and how those ontologies would be put together. Put differently, some people noticed the inferencing possibilities that can be obtained from moving from a representation in obo to one in OWL and others thought that some coordination among all those obo bio-ontologies would be advantageous given that post-hoc integration of ontologies of related and overlapping subject domains is not easy. Thus came into being the OBO Foundry to solve such issues, proposing an approach for coordinated evolution of ontologies to support biomedical data integration [Smith et al., 2007].

People in related disciplines, such as ecology, have taken on board experiences of these very early adopters, and instead decided to jump on board after the OWL standardization. They, however, were not only motivated by data(base) integration. Referring to Madin et al's paper [Madin et al., 2008], I highlight three points they made: "terminological ambiguity slows scientific progress, leads to redundant research efforts, and ultimately impedes advances towards a unified foundation for ecological science", i.e., identification of some serious problems they have in ecological research; "Formal ontologies provide a mechanism to address the drawbacks of terminological ambiguity in ecology", i.e., what they expect that ontologies will solve for them (disambiguation); and "and fill an important gap in the management of ecological data by facilitating powerful data discovery based on rigorously defined, scientifically meaningful terms", i.e., for what purpose they want to use ontologies and any associated computation (discovery using automated reasoning). That is, ontologies not as a—one of many possible—'*tool*' in the engineering infrastructure, but as a *required part of a method* in the scientific investigation that aims to discover new information and knowledge about nature (i.e., in answering the who, what, where, when, and how things are the way they are in nature). Success in inferring novel biological knowledge has been achieved with classification of protein phosphatases [Wolstencroft et al., 2007], precisely thanks to the expressive ontology and its automated reasoning services[17].

1.5 Outline and usage of the lecture notes

The preceding sections already indicated several aspect would return "later in the course". These lecture notes are ordered according to the course outline, but they do not necessarily have to be in that order.

The first option, that we will follow, is to commence with a recap of First Order Predicate Logic regarding the formalisation, and add the notion of *model-theoretic semantics* (what it all 'means', formally). Full FOL is undecidable, but there are less expressive languages, i.e., fragments of FOL, that are decidable[18] for a set of important problems in computing in the area of ontologies and knowledge bases. One such family of languages is the Description Logics (DL) family of languages. These two topics are covered in Chapter 2 and Chapter 3, respectively. Several DLs, in turn, form the basis of the W3C standardized Web Ontology Language OWL (actually, a family of languages, too).

[17]What they did, and how, will be described later in the course

[18]The principle advantage of having a decidable language to represent the knowledge, is that it is more amenable to computation and use in software

OWL specifies a computer-processable serialisation of the ontology and knowledge base, and interacts with the automated reasoners for OWL. OWL and the so-called standard reasoning services are summarised in Chapter 4.

After these logic foundations in Block I, we shall look at how one can develop an ontology. The first approach is a so-called 'top-down' approach, where we use foundational ontologies with the high-level categories and relationship to get us started with the principal choices and the modelling; this is covered in Chapter 6. However, designing an ontology from scratch is rather cumbersome, and much information already has been represented in various ways—natural language, conceptual data models, etc.—so, one can speed up ontology development also by somehow reusing those 'legacy' sources, which is described briefly in Chapter 7. Both approaches, however, are just that—not a 'cookbook recipe' for ontology development—and there exist interdependencies, methods, tools, and methodologies that help structure and carry out the activities, which is described in Chapter 5, therewith concluding Block II.

Blocks I and II form the foundations of ontology engineering (at an introductory level), and the topics that follow afterward deepen and extend that material. In one direction, you may wish to explore further some quite involved theory and technology to realise a practical ontology-driven information system, being querying databases by means of an ontology, which is the topic of Chapter 8. In a different direction are fancy extensions to the standard ontology languages, of which the temporal dimension is mentioned in Chapter 9.3, and uncertain & fuzzy in the extra topics in Chapter 9.

People not attending this module but who would like to go at their own pace through the material, also can start with Block II, and do Chapters 6, 7, and 5, or first Chapter 5 and then 6 and 7. Depending on one's background, one can study Chapters 2, 3, and 4, i.e., Block I, after Block II—unless one's knowledge of logic is a bit rusty or limited. In any case, both the material of Block I and Block II are prerequisites for Block III, advanced topics. Within Block III, Chapters 8, 9.3, and 9 can be done in order of preference.

This is the third version of the lecture notes, and due to time constraints, not everything that should have been in the notes made it into the notes (hence, don't take it as a textbook on ontologies at this stage). This is the case in particular at the instances where it mentions "we will cover this in more detail during the lecture"—which we will do, *and the lecture content and its slides are also part of the examinable course content.* If you do not attend the lectures and do not do the exercises, you are unlikely to pass the course.

As you will see, there are many references in the bibliography. You are not expected to read all of them; instead, each chapter has a "Literature and reference material" section with required and recommended reading. The large reference list may be useful especially for the practical assignment (Appendix A) and the mini-project (Appendix B): there are *very* many more references in computer science conference proceedings and journals, but the ones listed, first, in the "literature and reference material" and, second, in the bibliography, will give you a useful 'entry point' or may even suffice, depending on the chosen topics. As some of the scientific papers are a bit hard to read, or you may like to read up on some more general or more detailed material, I have listed several handbooks at the end of the bibliography, starting on page 166.

1.6 Exercises

The following exercises are of an exploratory nature.

1. There are several terms in the preceding sections that were highlighted in bold in the text. Find them, and describe them in your own words.

2. If you would like to get a practical 'feel' of ontologies and how they look like in an ontology development environment, you can run the Protégé ontology development environment installed on the lab PCs and play with the pizza ontology and its tutorial (see Vula for details) or load the AWO; if not now, you will have to do it later anyway.

3. In this year's instalment, we also will also set up and populate a Semantic Wiki. More detail about this will be communicated during the first lecture.

1.7 Literature and reference material

1. Berners-Lee, Tim; James Hendler and Ora Lassila. The Semantic Web. *Scientific American Magazine*, May 17, 2001. `http://www.sciam.com/article.cfm?id=the-semantic-web&print=true`
2. Nicola Guarino, Daniel Oberle, and Steffen Staab. What Is An Ontology? In: S. Staab and R. Studer, *Handbook on Ontologies*, Chapter 6. Springer. 2009. pp1-17.
3. Guarino, N. Formal Ontology in Information Systems. *Proceedings of FOIS'98*, Trento, Italy, June 6-8, 1998. IOS Press, Amsterdam, pp. 3-15

Part I

Logic foundations for ontologies

CHAPTER 2

First Order Logic recap

Although perhaps more foundations in ontologies is useful before delving into how to represent what you want to represent, having a basic grasp of logic-based ontology languages also can help understanding the ontologies and ontology engineering better. Therefore, we shall refresh the basics of first order logic in Section 2.1 (comprehensive introductions can be found elsewhere, e.g., [Hedman, 2004]), which is followed by a general idea of (automated) reasoning and two examples of tableau reasoning in Section 2.2.

2.1 First order logic syntax and semantics

Observe that logic is not the study of truth, but of the *relationship between the truth of one statement and that of another*. That is, in logic, we do not care whether a statement like "If angels exist then necessarily all of them fit on the head of a needle" (suitably formalised) is indeed true in reality[1], but if the if-part were true (resp., false), then what does that say about the truth value of the then-part of the statement? And likewise for a whole bunch of such sentences. Others do care what is represented formally with such a logic language, but we will defer that to Block II of the course.

To be able to study those aspects of logic, we need a language that is unambiguous; natural language is not. You may have encountered propositional logic already, and first order predicate logic (FOL) is an 'extension' of that, which enables us to represent more knowledge in more detail. Here, I will give only a brief glimpse of it. Eventually, you will need to be able to recognise, understand, and be able to formalise at least a little bit in FOL. Of all the definitions that will follow shortly, there are four important ideas to grasp: the *syntax* of a language, the *model-theoretic semantics* of a language, what a *theory* means in the context of logic, and the notion of *deduction* where we apply some rules to what we have represented to derive knowledge that was implicitly represented.

First, there are two principal components to consider for the language:

- **Syntax**, which has to do with what 'things' (symbols, notations) can we use in the language; there is/are a(n):
  - Alphabet
  - Languages constructs
  - Sentences to assert knowledge

[1]or, for that matter, whether there is a reality and whether we have access to it

- **Semantics**
  - Formal meaning, which has to do what those sentences with the alphabet and constructs actual are supposed to mean.

2.1.1 Syntax

The lexicon of a first order language contains the following:

- Connectives and Parentheses: $\neg$, $\rightarrow$, $\leftrightarrow$, $\wedge$, $\vee$, (and);
- Quantifiers: $\forall$ (universal) and $\exists$ (existential);
- Variables: $x, y, z, \ldots$ ranging over particulars (individual objects);
- Constants: $a, b, c, \ldots$ representing a specific element;
- Functions: $f, g, h, \ldots$, with arguments listed as $f(x_1, \ldots x_n)$;
- Relations: $R, S, \ldots$ with an associated arity.

There is an (countably infinite) supply of *symbols* (signature): Variables, Functions, Constants, and Relations.

In other words: we can use these things to create 'sentences', like we have in natural language, but then controlled and with a few extra figurines. Let us look first at how we can formalise a natural language sentence into first order logic.

Example 2.1. From Natural Language to First order logic (or vv.) Consider the following three sentences:

- "Each animal is an organism"
- "All animals are organisms"
- "If it is an animal then it is an organism"

This can be formalised as:

$$\forall x(Animal(x) \rightarrow Organism(x)) \tag{2.1}$$

(there's some explanation about this in the slides). Instead of talking about all objects of a particular type, we also can assert there are at least some of them; e.g.,

- "Aliens exist"

could be formalised as

$$\exists x \; Alien(x) \tag{2.2}$$

and

- "There are books that are heavy"

(well, at least one of them is) as:

$$\exists x(Book(x) \wedge heavy(x)) \tag{2.3}$$

A sentence—or, more precisely, its meaning—such as "Each student must be registered for a degree programme" requires a bit more consideration. There are at least two ways to say the same thing (we leave the arguments for and against each option for another time):

i) $\forall x, y(registered\_for(x, y) \rightarrow Student(x) \wedge DegreeProgramme(y))$
   "if there is a *registered_for* relation, then the first object is a student and the second one a degree programme"
   $\forall x(Student(x) \rightarrow \exists y \; registered\_for(x, y))$
   "Each student is registered for at least one y", where the y is a degree programme (taken from the first axiom)

ii) $\forall x(Student(x) \rightarrow \exists y \; (registered\_for(x, y) \wedge DegreeProgramme(y)))$
   "'Each student is registered for at least one degree programme'

But all this is still just syntax (it does not say what it really means), and it looks like a 'free for all' on how we can use these symbols. This is, in fact, not the case, and the remainder of the definitions will make this more precise, which will be illustrated in Example 2.2 afterward. ♢

Terms: A term is inductively defined by two rules, being:

- Every variable and constant is a term.
- if f is a m-ary function and $t_1, \ldots t_m$ are terms, then $f(t_1, \ldots, t_m)$ is also a term.

Definition 2.1. (Atomic formula) *An* atomic formula *is a formula that has the form $t_1 = t_2$ or $R(t_1, ..., t_n)$ where R is an n-ary relation and $t_1, ..., t_n$ are terms.*

Definition 2.2. (Formula) *A string of symbols is a* formula *of FOL if and only if it is constructed from atomic formulas by repeated applications of rules R1, R2, and R3.*

R1. If ϕ is a formula then so is $\neg\phi$.
R2. If ϕ and ψ are formulas then so is $\phi \wedge \psi$.
R3. If ϕ is a formula then so is $\exists x \phi$ for any variable x.

A *free variable* of a formula ϕ is that variable occurring in ϕ that is not quantified. We then can introduce the definition of *sentence.*

Definition 2.3. (Sentence) *A* sentence *of FOL is a formula having no free variables.*

2.1.2 Semantics

Whether a sentence is true or not depends on the underlying set and the interpretation of the function, constant, and relation symbols. To this end, we have structures: a *structure* consists of an *underlying set* together with an *interpretation* of functions, constants, and relations. Given a sentence ϕ and a structure M, M *models* ϕ means that the sentence ϕ is true with respect to M. More precisely,

Definition 2.4. (Vocabulary) *A* vocabulary $\mathcal{V}$ *is a set of function, relation, and constant symbols.*

Definition 2.5. ($\mathcal{V}$-structure) *A* $\mathcal{V}$-structure *consists of a non-empty underlying set Δ along with an interpretation of $\mathcal{V}$. An interpretation of $\mathcal{V}$ assigns an element of Δ to each constant in $\mathcal{V}$, a function from Δ^n to Δ to each n-ary function in $\mathcal{V}$, and a subset of Δ^n to each n-ary relation in $\mathcal{V}$. We say M is a* structure *if it is a $\mathcal{V}$-structure of some vocabulary $\mathcal{V}$.*

Definition 2.6. ($\mathcal{V}$-formula) *Let $\mathcal{V}$ be a vocabulary. A* $\mathcal{V}$-formula *is a formula in which every function, relation, and constant is in $\mathcal{V}$. A* $\mathcal{V}$-sentence *is a $\mathcal{V}$-formula that is a sentence.*

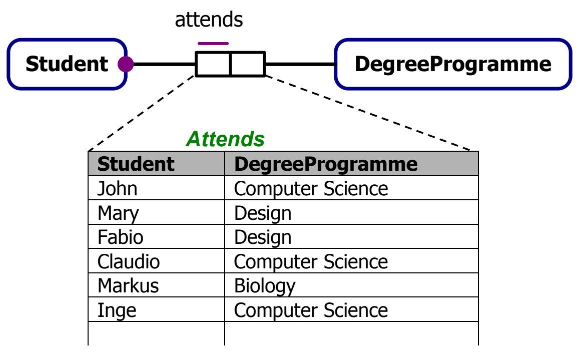
Note: When we say that M *models* ϕ, denoted with $M \models \phi$, this is with respect to M being a $\mathcal{V}$-structure and $\mathcal{V}$-sentence ϕ is true in M.

Model theory: the interplay between M and a set of first-order sentences $\mathcal{T}(M)$, which is called the *theory of* M, and its 'inverse' from a set of sentences Γ to a class of structures.

Definition 2.7. (Theory of M) *For any $\mathcal{V}$-structure M, the* theory of M*, denoted with $\mathcal{T}(M)$, is the set of all $\mathcal{V}$-sentences ϕ such that $M \models \phi$.*

Student **is an entity type**.
DegreeProgramme **is an entity type**.
Student attends DegreeProgramme.

Each Student attends **exactly one** DegreeProgramme.
It is possible that more than one Student attends **the same** DegreeProgramme.
OR, in the negative:
For each Student, **it is impossible that that** Student attends **more than one** DegreeProgramme.
It is impossible that any Student attends **no** DegreeProgramme.



Student	DegreeProgramme
John	Computer Science
Mary	Design
Fabio	Design
Claudio	Computer Science
Markus	Biology
Inge	Computer Science

Figure 2.1: A theory denoted in ORM notation, ORM verbalization, and some data in the database. See Example 2.2 for details.

Definition 2.8. (Model) *For any set of $\mathcal{V}$-sentences, a* model *of* Γ *is a* $\mathcal{V}$*-structure that models each sentence in* Γ*. The class of all models of* Γ *is denoted by* $\mathcal{M}(\Gamma)$*.*

Now we can go to the interesting notions: *theory* in the context of logic:

Definition 2.9. (Complete $\mathcal{V}$-theory) *Let* Γ *be a set of* $\mathcal{V}$*-sentences. Then* Γ *is a* complete $\mathcal{V}$-theory *if, for any* $\mathcal{V}$*-sentence* ϕ *either* ϕ *or* $\neg\phi$ *is in* Γ *and it is not the case that both* ϕ *and* $\neg\phi$ *are in* Γ*.*

It can then be shown that for any $\mathcal{V}$-structure M, $\mathcal{T}(M)$ is a complete $\mathcal{V}$-theory (for proof, see *e.g.* [Hedman, 2004], p90).

Definition 2.10. *A set of sentences* Γ *is said to be* consistent *if no contradiction can be derived from* Γ*.*

Definition 2.11. (Theory) *A* theory *is a consistent set of sentences.*

The latter two definitions are particularly relevant later on when we look at the typical reasoning services for ontologies.

Example 2.2. How does this work out in practice? Let us take something quasi-familiar: a conceptual data model in Object-Role Modeling notation, depicted in the bottom-half of Figure 2.1 (the top-half is its 'verbalisation' in a controlled natural language).

First, we consider it as a theory, creating a logical reconstruction of the icons in the figure. There is one binary predicate, attends, and there are two unary predicates, Student and DegreeProgramme. The binary predicate is typed, i.e., its domain and range are defined, hence:

$$\forall x, y(attends(x, y) \rightarrow Student(x) \land DegreeProgramme(y)) \tag{2.4}$$

Note that x and y quantify over the whole axiom (thanks to the brackets), hence, there are no free variables, hence, it is a sentence. There are two constraints in the figure: the blob and the line over part of the rectangle, and, textually, "Each Student attends exactly one DegreeProgramme" and "It is possible that more than one Student attends the same DegreeProgramme". The first constraint can be formalised (in short-hand notation):

$$\forall x(Student(x) \rightarrow \exists^{=1} y\ attends(x, y)) \tag{2.5}$$

So, our vocabulary is $\{attends, Student, DegreeProgramme\}$, and we have two sentences (Eq. 2.4 and Eq. 2.5). The sentences form the theory, as they are not contradicting and admit a model.
Let us now consider the structure. We have a non-empty underlying set:
$\Delta = \{John, Mary, Fabio, Claudia, Markus, Inge, ComputerScience, Biology, Design\}$.
The interpretation then maps the instances in Δ with the elements in our vocabulary; that is, we end up with $\{John, Mary, Fabio, Claudio, Markus, Inge\}$ as objects in $Student$, and likewise for $DegreeProgramme$ and the binary $attends$. Observe that this structure does not contradict the constraints of our sentences. ♢

Equivalences With the syntax and semantics, several equivalencies between formulae can be proven. We list them for easy reference, with a few 'informal reading' for illustration. ϕ, ψ, and χ are formulas.

- Commutativity:
  $\phi \wedge \psi \equiv \psi \wedge \phi$
  $\phi \vee \psi \equiv \psi \vee \phi$
  $\phi \leftrightarrow \psi \equiv \psi \leftrightarrow \phi$
- Associativity:
  $(\phi \wedge \psi) \wedge \chi \equiv \phi \wedge (\psi \wedge \chi)$
  $(\phi \vee \psi) \vee \chi \equiv \phi \vee (\psi \vee \chi)$ *// just like it doesn't matter for subtraction*
- Idempotence:
  $\phi \wedge \phi \equiv \phi$
  $\phi \vee \phi \equiv \phi$ *//'itself or itself is itself'*
- Absorption:
  $\phi \wedge (\phi \vee \psi) \equiv \phi$
  $\phi \vee (\phi \wedge \psi) \equiv \phi$
- Distributivity:
  $(\phi \vee (\psi \wedge \chi) \equiv (\phi \vee \psi) \wedge (\phi \vee \chi)$
  $(\phi \wedge (\psi \vee \chi) \equiv (\phi \wedge \psi) \vee (\phi \wedge \chi)$
- Double negation:
  $\neg\neg\phi \equiv \phi$
- De Morgan:
  $\neg(\phi \wedge \psi) \equiv \neg\phi \vee \neg\psi$
  $\neg(\phi \vee \psi) \equiv \neg\phi \wedge \neg\psi$ *//'negation of a disjunction implies the negation of each of the disjuncts'*
- Implication:
  $\phi \rightarrow \psi \equiv \neg\phi \vee \psi$
- Tautology:
  $\phi \vee \top \equiv \top$
- Unsatisfiability:
  $\phi \wedge \bot \equiv \bot$
- Negation:
  $\phi \wedge \neg\phi \equiv \bot$ *//something cannot be both true and false*
  $\phi \vee \neg\phi \equiv \top$

- Neutrality:
  $\phi \land \top \equiv \phi$
  $\phi \lor \bot \equiv \phi$
- Quantifiers:
  $\neg\forall x.\phi \equiv \exists x.\neg\phi$
  $\neg\exists x.\phi \equiv \forall x.\neg\phi$ *//'if there does not exist some, then there's always none'*
  $\forall x.\phi \land \forall x.\psi \equiv \forall x.(\phi \land \psi)$
  $\exists x.\phi \lor \exists x.\psi \equiv \exists x.(\phi \lor \psi)$
  $(\forall x.\phi) \land \psi \equiv \forall x.(\phi \land \psi)$ if x is not free in ψ
  $(\forall x.\phi) \lor \psi \equiv \forall x.(\phi \lor \psi)$ if x is not free in ψ
  $(\exists x.\phi) \land \psi \equiv \exists x.(\phi \land \psi)$ if x is not free in ψ
  $(\exists x.\phi) \lor \psi \equiv \exists x.(\phi \lor \psi)$ if x is not free in ψ

Note: The ones up to (but excluding) the quantifiers hold for both propositional logic and first order predicate logic.

2.2 Reasoning

Having a logic language with a semantics is one thing, but, oftentimes, it is a means to an end rather than an end in itself. From the computational angle—especially from a logician's perspective—the really interesting aspect of having such a language and (someone else) having gone through formalising some subject domain, is to reason over it. Here, we are not talking about making a truth table, which is way too costly when one has to analyse many sentences, but deploying other techniques so that it can be scaled up compared to the manual efforts. Automated reasoning, then, concerns computing systems that automate the ability to make inferences by designing a formal language in which a problem's assumptions and conclusion can be written and providing correct algorithms to solve the problem with a computer in an efficient way.

To this end some terminology has to be introduced first, using the notion of "formula" that was introduced in the previous section:

- A formula is *valid* if it holds under *every* assignment. $\models \phi$ to denote this. A valid formula is called a *tautology*.
- A formula is *satisfiable* if it holds under *some* assignment.
- A formula is *unsatisfiable* if it holds under *no* assignment. An unsatisfiable formula is called a *contradiction*.

How does one find out whether a formula is valid or not? How do we find out whether our knowledge base is satisfiable? The main proof technique for DL-based ontologies is tableaux, although there are several others. (Note: the remainder of Section 2.2 are amended versions of my "Reasoning, automated" essay and related definitions that has been published in Springer's Encyclopedia of Systems Biology.)

2.2.1 Introduction

Characteristics

People employ reasoning informally by taking a set of premises and somehow arriving at a conclusion, i.e., it is *entailed by* the premises (deduction), arriving at a hypothesis (abduction), or generalizing from facts to an assumption (induction). Mathematicians and computer scientists developed ways to capture this formally with logic languages to represent the knowledge and rules that may be applied to the axioms so that one can construct a *formal proof* that the conclusion can be derived from the premises. This can be done by hand [Solow, 2005] for small theories, but this does not scale up when one has, say, 80 or more axioms even though there are much larger theories that require a formal analysis, such as checking that the theory can indeed have a model and thus does not contradict itself. To this end, much work has gone into automating reasoning. The remainder of this section introduces briefly several of the many purposes and usages of automated reasoning, its limitations, and types of automated reasoners.

Purposes

Automated reasoning, and deduction in particular, has found applications in 'every day life'. A notable example is hardware and (critical) software verification, which gained prominence after Intel had shipped its Pentium processors with a floating point unit error in 1994 that lost the company about $500 million. Since then, chips are routinely automatically proven to function correctly according to specification before taken into production. A different scenario is scheduling problems at schools to find an optimal combination of course, lecturer, and timing for the class or degree program, which used to take a summer to do manually, but can now be computed in a fraction of it using constraint programming. In addition to such general applications domains, it is also used for specific scenarios, such as the demonstration of discovering (more precisely: deriving) novel knowledge about protein phosphatases by Wolstencroft and coauthors [Wolstencroft et al., 2007]. They represented the knowledge about the subject domain of protein phosphatases in humans in a formal bio-ontology and classified the enzymes of both human and the fungus *Aspergillus fumigatus* using an automated reasoner, which showed that (i) the reasoner was as good as human expert classification, (ii) it identified additional p-domains (an aspect of the phosphatases) so that the human-originated classification could be refined, and (iii) it identified a novel type of calcineurin phosphatase like in other pathogenic fungi. The fact that one can use an automated reasoner (in this case: deduction, using a Description Logics knowledge base) as a viable method in science is an encouragement to explore such avenues further.

Limitations

While many advances have been made in specific application areas, the main limitation of the implementations are due to the computational complexity of the chosen representation language and the desired automated reasoning services. This is being addressed by implementations of optimizations of the algorithms or by limiting the expressiveness of the language, or both. One family of logics that focus principally on computationally well-behaved languages is Description Logics, which are decidable fragments of first order logic [Baader et al., 2008]; that is, they are languages such that the corresponding reasoning services are guaranteed to terminate with an answer. Description Logics form the basis of most of the Web Ontology Languages OWL and OWL 2 and are gaining increasing importance in the Semantic Web applications area. Giving up expressiveness, however, does lead to criticism from the modelers community, as a computationally nice language may not have the features deemed necessary to represent the subject domain adequately.

Tools

There are many tools for automated reasoning, which differ in which language they accept, the reasoning services they provide, and, with that, the purpose they aim to serve.

There are, among others, generic first- and higher order logic theorem provers (e.g., Prover9, MACE4, Vampire, HOL4), SAT solvers that compute if there is a model for the formal theory (e.g., GRASP, Satz), Constraint Satisfaction Programming for solving, e.g., scheduling problems and reasoning with soft constraints (e.g., Eclipse), DL reasoners that are used for reasoning over OWL ontologies using deductive reasoning to compute satisfiability, consistency, and perform taxonomic and instance classification (e.g., Fact++, RacerPro, Hermit, CEL, QuOnto), and inductive logic programming tools (e.g., PROGOL and Aleph).

2.2.2 Basic idea

Essential to automated reasoning are

i. *The choice of the class of problems the software program has to solve*, such as checking the consistency of a theory (i.e., there are no contradictions) or computing a classification hierarchy of concepts subsuming one another based on the properties represented in the logical theory;
ii. *The formal language in which to represent the problems*, which may have more or less features to represent the subject domain knowledge, such as cardinality constraints (e.g., a member of the Arachnids has as part exactly eight legs), probabilities, or temporal knowledge (e.g., that a butterfly is a transformation of a caterpillar);
iii. *The way how the program has to compute the solution*, such as using natural deduction or resolution; and
iv. *How to do this efficiently*, be this achieved through constraining the language into one of low complexity, or optimizing the algorithms to compute the solution, or both.

Concerning the first item, with a *problem* being, e.g., "is my theory is consistent?", then the *problem's assumptions* are the axioms in the logical theory and the *problem's conclusion* that is computed by the automated reasoner is a "yes" or a "no" (provided the language in which the assumptions are represented is decidable and thus guaranteed to terminate with an answer). With respect to how this is done (item iii), two properties are important for the calculus used: soundness and completeness ($T \vdash \psi$ if and only if $M \models \psi$). If it is incomplete, then there exist entailments that cannot be computed (hence, 'missing' some results), if it is unsound then false conclusions can be derived from true premises, which his even more undesirable. An example is included in Section 2.2.4 proving the validity of a formula (the class of the problem) in propositional logic (the formal language) using tableau reasoning (the way how to compute the solution) with the Tree Proof Generator[2] (the automated reasoner), and more detail, with reflection, and other techniques can be found in, among others, [Portoraro, 2010].

2.2.3 Deduction, Abduction, Induction

Deduction

Deduction is a way to ascertain if a theory T represented in a logic language entails an axiom α that is not explicitly asserted in T (written as $T \models \alpha$), i.e., α can be *derived* from the premises through repeated application of *deduction rules*. For instance, a theory that states that "each Arachnid has as part 8 legs" and "each Tarantula is an Arachnid" then one can deduce—it is entailed in the theory—that "Each Tarantula has as part 8 legs".

[2] `http://www.umsu.de/logik/trees/`

An example is included below, which formally demonstrates that a formula is entailed in a theory T using said rules.

Thus, strictly speaking, a deduction does not reveal *novel* knowledge, but only that what was already represented implicitly in the theory. Nevertheless, with large theories, it is often difficult to oversee all implications of the represented knowledge and, hence, the deductions may be perceived as novel from a domain expert perspective, such as with the example about the protein phosphatases. (This is in contrast to Abduction and Induction, where the reasoner 'guesses' knowledge that is not already entailed in the theory.)

Mechanisms for deductions There are various ways how to ascertain $T \models \alpha$, be it manually or automatically. One can construct a step-by-step proof 'forward' from the premises by applying the deduction rules or prove it indirectly such that $T \cup \{\neg\alpha\}$ must lead to a contradiction. The former approach is called *natural deduction*, whereas the latter is based on techniques such as resolution, matrix connection methods, and sequent deduction (which includes *tableaux*).

Concerning deduction rules for tableaux and first order predicate logic formulae, we have, as with the example in Section 2.2.4, the two deduction rules that if a model satisfies a conjunction, then it also satisfies each of the conjuncts, which is written as follows:

$$\frac{\phi \land \psi}{\begin{array}{c}\phi\\ \psi\end{array}}$$

and if a model satisfies a disjunction, then it also satisfies one of the disjuncts

$$\frac{\phi \lor \psi}{\phi \mid \psi}$$

In addition, there are two rules for the quantified formulas. First, if a model satisfies a universally quantified formula ($\forall$), then it also satisfies the formula where the quantified variable has been substituted with some term (and the prescription is to use all the terms which appear in the tableaux),

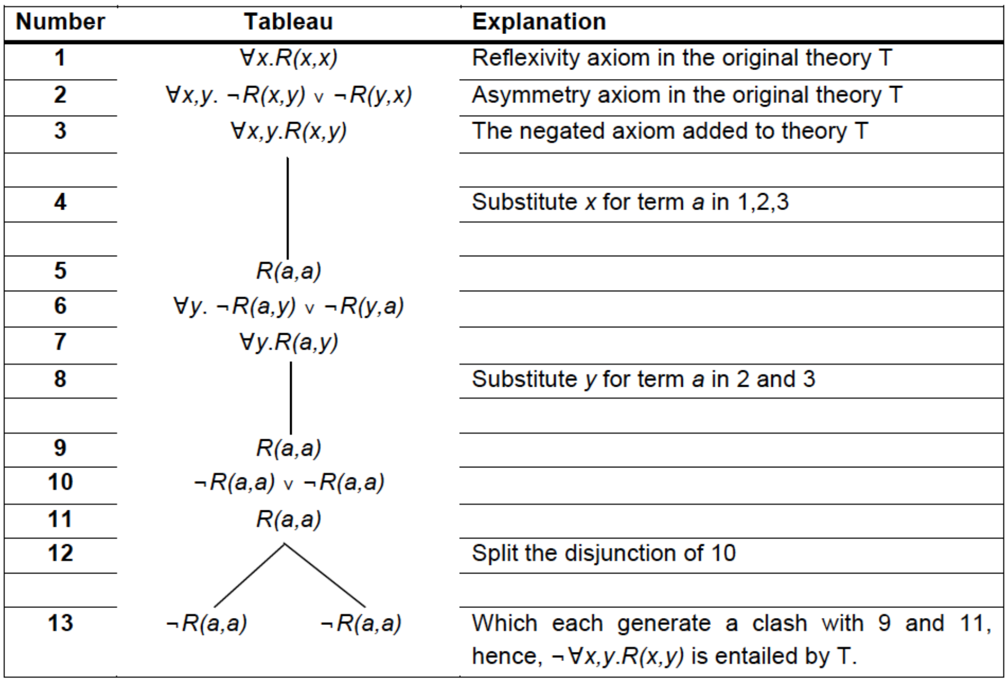
$$\frac{\forall x.\phi}{\begin{array}{l}\phi\{X/t\}\\ \forall x.\phi\end{array}}$$

and, second, for an existentially quantified formula, if a model satisfies it, then it also satisfies the formula where the quantified variable has been substituted with a new Skolem constant,

$$\frac{\exists x.\phi}{\phi\{X/a\}}$$

Example 2.3. Let us take some arbitrary theory T that contains two axioms stating that relation R is reflexive ($\forall x(R(x,x))$, a thing relates to itself) and asymmetric ($\forall x, y(R(x,y) \rightarrow \neg R(y,x))$, if a thing a relates to b through relation R, then b does not relate back to a). We then can deduce, among others, that $T \cup \{\neg\forall x, y(R(x,y))\}$ is satisfiable. We do this by demonstrating that the negation of the axiom is unsatisfiable.

To enter the tableau, we first rewrite the asymmetry into a disjunction using equivalences, i.e., $\forall x, y(R(x,y) \rightarrow \neg R(y,x))$ is equivalent to $\forall x, y(\neg R(x,y) \lor R(y,x))$, and add a negation to $\{\neg\forall x, y(R(x,y))\}$, which thus becomes $\forall x, y(R(x,y))$. Then, to start the tableau, we have three axioms (1, 2, 3) and the full tableau as in Figure 2.2. $\diamondsuit$



Number	Tableau	Explanation
1	$\forall x.R(x,x)$	Reflexivity axiom in the original theory T
2	$\forall x,y.\ \neg R(x,y) \vee \neg R(y,x)$	Asymmetry axiom in the original theory T
3	$\forall x,y.R(x,y)$	The negated axiom added to theory T
4		Substitute *x* for term *a* in 1,2,3
5	$R(a,a)$	
6	$\forall y.\ \neg R(a,y) \vee \neg R(y,a)$	
7	$\forall y.R(a,y)$	
8		Substitute *y* for term *a* in 2 and 3
9	$R(a,a)$	
10	$\neg R(a,a) \vee \neg R(a,a)$	
11	$R(a,a)$	
12		Split the disjunction of 10
13	$\neg R(a,a)$ $\neg R(a,a)$	Which each generate a clash with 9 and 11, hence, $\neg\forall x,y.R(x,y)$ is entailed by T.

Figure 2.2: Tableau example (using notation with a "." not the brackets).

Abduction

Compared to deduction, there is less permeation of automated reasoning for abduction. From a scientist's perspective, automation of abduction may seem appealing, because it would help one generate a hypothesis based on the facts put into the reasoner [Aliseda, 2004]. Practically, it has been used for, for instance, fault detection: given the knowledge about a system and the observed defective state, find the likely fault in the system. To formally capture theory with assumptions and facts and find the conclusion, several approaches have been proposed, each with their specific application areas; for instance, sequent calculus, belief revision, probabilistic abductive reasoning, and Bayesian networks.

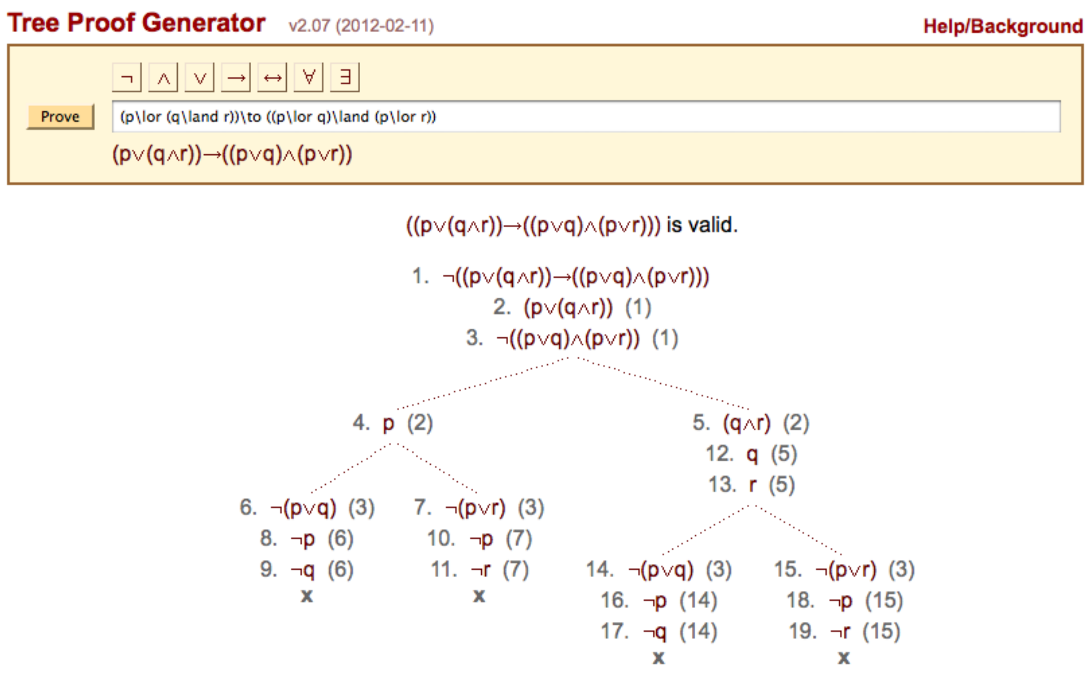
Induction

Induction allows one to arrive at a conclusion that actually may be false even though the premises are true. The premises provide a *degree of support* so as to infer a as an explanation of b. Such a 'degree' can be based on probabilities (a statistical syllogism) or analogy. For instance, we have a premise that "The proportion of bacteria that acquire genes through horizontal gene transfer is 95%" and the fact that "*Staphylococcus aureus* is a bacterium", then we induce that the probability that *S. aureus* acquires genes through horizontal gene transfer is 95%. Induction by analogy is weaker version of reasoning, in particular in logic-based systems, and yields very different answers than deduction. For instance, let us encode that some instance, Tibbles, is a cat and we know that all cats have the properties of having a tail, four legs, and are furry. When we encode that another animal, Tib, who happens to have four legs and is also furry, then by inductive reasoning by analogy, we conclude that Tib is also a cat, even though in reality it may well be an instance of cheetah. On the other hand, by deductive reasoning, Tib will not be classified as being an instance of cat (but may be an instance of a superclass of cats (e.g., still within the suborder *Feliformia*), provided that the superclass has declared that all instances have four legs and are furry. Given that humans do perform such reasoning, there are attempts to mimic this process in software applications, most notably in the area of machine learning and inductive logic programming. The principal approach with inductive logic programming is to take as input positive examples + negative examples + background knowledge and then derive a hypothesized logic program that entails all the positive and none of the negative examples.

2.2.4 Proof

A proof is a convincing argument expressed in the language of mathematics.

Example 2.4. A sample computation to prove automatically whether the propositional formula $((p \vee (q \wedge r)) \rightarrow ((p \vee q) \wedge (p \vee r))$ is valid or not is included in Figure 2.3 and Figure 2.4, using tableau reasoning (see Deduction, Section 2.2.3). The tableau method is a decision procedure that checks the existence of a model (i.e., that it can be instantiated). It exhaustively looks at all the possibilities, so that it can eventually prove that *no* model could be found for *unsatisfiable* formulas (if it is satisfiable, we have found a counterexample). This is done by decomposing the formula in top-down fashion after it has been translated into Negation Normal Form (i.e., all the negations have been pushed inside), which can be achieved using equivalences. Further, if a model satisfies a conjunction, then it also satisfies each of the conjuncts ("$\wedge$"), and if a model satisfies a disjunction ("$\vee$"), then it also satisfies one of the disjuncts (this is a non-deterministic rule and it generates two alternative branches). Last, one has to apply these completion rules until either (a) an explicit contradiction is obtained due to the presence of two opposite literals in a node (a clash) is generated in each branch, or (b) there is a completed branch where no more rule is applicable. $\diamondsuit$



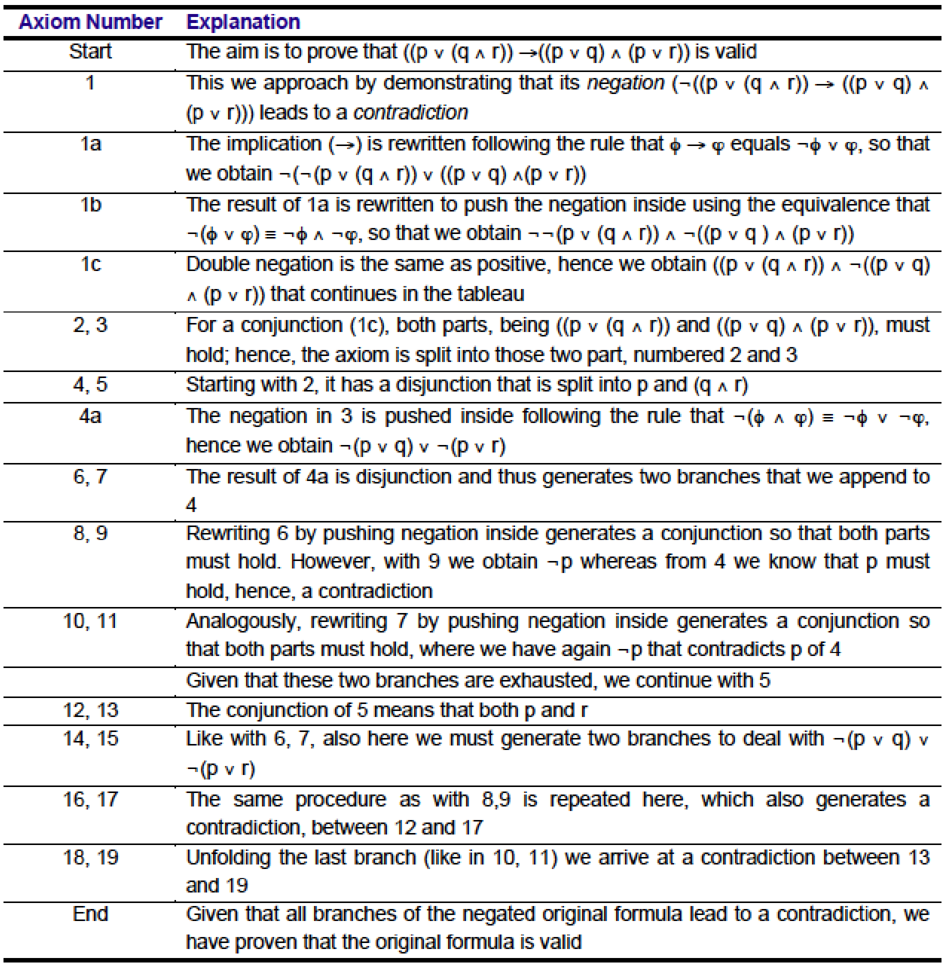
```
(p\lor (q\land r))\to ((p\lor q)\land (p\lor r))
(p∨(q∧r))→((p∨q)∧(p∨r))

((p∨(q∧r))→((p∨q)∧(p∨r))) is valid.

1. ¬((p∨(q∧r))→((p∨q)∧(p∨r)))
2. (p∨(q∧r)) (1)
3. ¬((p∨q)∧(p∨r)) (1)

4. p (2)                              5. (q∧r) (2)
                                      12. q (5)
                                      13. r (5)
6. ¬(p∨q) (3)    7. ¬(p∨r) (3)      14. ¬(p∨q) (3)   15. ¬(p∨r) (3)
8. ¬p (6)        10. ¬p (7)          16. ¬p (14)      18. ¬p (15)
9. ¬q (6)        11. ¬r (7)          17. ¬q (14)      19. ¬r (15)
x                x                   x                x
```

Figure 2.3: Sample computation using semantic tableau proving that the propositional formula is valid; see Figure 2.4 for an explanation.



Axiom Number	Explanation
Start	The aim is to prove that ((p ∨ (q ∧ r)) →((p ∨ q) ∧ (p ∨ r)) is valid
1	This we approach by demonstrating that its *negation* (¬((p ∨ (q ∧ r)) → ((p ∨ q) ∧ (p ∨ r))) leads to a *contradiction*
1a	The implication (→) is rewritten following the rule that ϕ → φ equals ¬ϕ ∨ φ, so that we obtain ¬(¬(p ∨ (q ∧ r)) ∨ ((p ∨ q) ∧(p ∨ r))
1b	The result of 1a is rewritten to push the negation inside using the equivalence that ¬(ϕ ∨ φ) ≡ ¬ϕ ∧ ¬φ, so that we obtain ¬¬(p ∨ (q ∧ r)) ∧ ¬((p ∨ q) ∧ (p ∨ r))
1c	Double negation is the same as positive, hence we obtain ((p ∨ (q ∧ r)) ∧ ¬((p ∨ q) ∧ (p ∨ r)) that continues in the tableau
2, 3	For a conjunction (1c), both parts, being ((p ∨ (q ∧ r)) and ((p ∨ q) ∧ (p ∨ r)), must hold; hence, the axiom is split into those two part, numbered 2 and 3
4, 5	Starting with 2, it has a disjunction that is split into p and (q ∧ r)
4a	The negation in 3 is pushed inside following the rule that ¬(ϕ ∧ φ) ≡ ¬ϕ ∨ ¬φ, hence we obtain ¬(p ∨ q) ∨ ¬(p ∨ r)
6, 7	The result of 4a is disjunction and thus generates two branches that we append to 4
8, 9	Rewriting 6 by pushing negation inside generates a conjunction so that both parts must hold. However, with 9 we obtain ¬p whereas from 4 we know that p must hold, hence, a contradiction
10, 11	Analogously, rewriting 7 by pushing negation inside generates a conjunction so that both parts must hold, where we have again ¬p that contradicts p of 4
	Given that these two branches are exhausted, we continue with 5
12, 13	The conjunction of 5 means that both p and r
14, 15	Like with 6, 7, also here we must generate two branches to deal with ¬(p ∨ q) ∨ ¬(p ∨ r)
16, 17	The same procedure as with 8,9 is repeated here, which also generates a contradiction, between 12 and 17
18, 19	Unfolding the last branch (like in 10, 11) we arrive at a contradiction between 13 and 19
End	Given that all branches of the negated original formula lead to a contradiction, we have proven that the original formula is valid

Figure 2.4: Explanation of the tableaux in Figure 2.3.

2.3 Exercises

Exercise 1. This exercise is to refresh your memory if you know propositional logic. Is the following argument valid, a tautology, a contradiction, satisfiable, or neither? Represent the argument formally in propositional logic and use truth tables to prove it.

Dalila travels to Johannesburg or she travels to Durban.
If she travels to Johannesburg, she takes the plane.
Therefore, Dalila does not travel to Durban.

Exercise 2. Consider the structures in Figure 2.5, which are graphs.

a. Figures 2.5-A and B are different depictions, but have the same descriptions w.r.t. the vertices and edges. Check this.
b. C has a property that A and B do not have. Represent this in a first-order sentence.
c. Find a suitable first-order language for A (/B), and formulate at least two properties of the graph using quantifiers.

Exercise 3. Consider the graph in Figure 2.5, and first-order language $\mathcal{L} = \langle R \rangle$, with R being a binary relation symbol (edge).

a. Formalise the following properties of the graph as $\mathcal{L}$-sentences: (i) (a, a) and (b, b) are edges of the graph; (ii) (a, b) is an edge of the graph; (iii) (b, a) is not an edge of the graph. Let T stand for the resulting set of sentences.

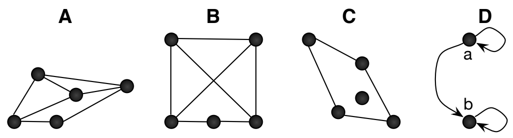


Figure 2.5: Graphs for Exercise 2 (figures A-C) and Exercise 3 (figure D).

b. Prove that $T \cup \{\forall x \forall y R(x, y)\}$ is unsatisfiable using tableaux calculus.

Exercise 4. Let us have a logical theory Θ with the following sentences:

- $\forall x Pizza(x)$, $\forall x PizzaT(x)$, $\forall x PizzaB(x)$, which are disjoint
- $\forall x(Pizza(x) \rightarrow \neg PizzaT(x))$,
- $\forall x(Pizza(x) \rightarrow \neg PizzaB(x))$,
- $\forall x(PizzaT(x) \rightarrow \neg PizzaB(x))$,
- $\forall x, y(hasT(x, y) \rightarrow Pizza(x) \wedge PizzaT(y))$,
- $\forall x, y(hasB(x, y) \rightarrow Pizza(x) \wedge PizzaB(y))$,
- $\forall x(ITPizza(x) \rightarrow Pizza(x))$, and
- $\forall x(ITPizza(x) \rightarrow \neg \exists y(hasT(x, y) \wedge FruitT(y))$, where
- $\forall x(VegeT(x) \rightarrow PizzaT(x))$ and
- $\forall x(FruitT(x) \rightarrow PizzaT(x))$.

Task:

a. A Pizza margherita has the necessary and sufficient conditions that it has mozzarella, tomato, basilicum and oil as toppings and has a pizza base. Add this to Θ. Annotate you commitments: what have you added to Θ and how?
   Hint: fruits are not vegetables, categorise the toppings, and "necessary and sufficient" is denoted with $\leftrightarrow$.
b. We want to merge our new Θ with some other theory Γ that has knowledge about fruits and vegetables. Γ contains, among other formulas, $\forall x(Tomato(x) \rightarrow Fruit(x))$. What happens? Represent the scenario formally, and prove your answer.

Actually, this is not easy to figure out manually, and there are ways to automate this, which you will do later in Chapter 4.

2.4 Literature and reference material

The following literature is optional for this course

1. Hedman, S. *A first course in logic—an introduction to model theory, proof theory, computability, and complexity.* Oxford: Oxford University Press. 2004.
2. Solow, D. How to read and do proofs. 4th Ed. Wiley. 2005.

CHAPTER 3

Description Logics

A Description Logic (DL) is a structured fragment of FOL; more precisely: any (basic) Description Logic language is a subset of $\mathcal{L}_3$, i.e., the function-free FOL using only at most three variable names, and its representation is at the predicate level: no variables are present in the formalism. DLs provide a logical reconstruction and (claimed to be a) unifying formalism for other knowledge representation languages, such as frames-based systems, object-oriented modelling, Semantic data models, etc. They provide the language to formulate theories and systems declaratively *expressing structured information* and for *accessing* and *reasoning* with it, and they are used for, among others, terminologies and ontologies, formal conceptual data modelling, and information integration.

Figure 3.1 shows a basic overview of the principal components of a DL knowledge base, with the so-called *TBox* containing the knowledge at the class-level and the *ABox* containing the data (individuals). Sometimes you will see added to the figure an *RBox*, which is used to make explicit there are relationships and the axioms that hold for them.

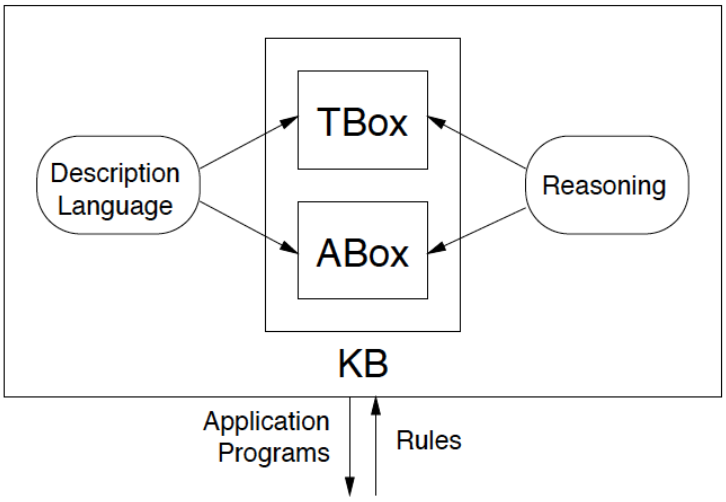


Figure 3.1: A Description Logic knowledge base (Source: [Baader et al., 2008]).

The remainder of this section contains, first, a general introduction to DL (Section 3.1), which are the first five sections of the DL Primer [Krötzsch et al., 2012], and is reproduced here with permission of its authors Markus Krötzsch, František Simančík, and Ian Horrocks. This is followed by a shorter notation style for the DL language $\mathcal{ALC}$ in Section 3.2, which one encounters typically in the literature (based on Section 3.1, this should be understandable, and it will pass the revue during the lectures); slightly more detailed introductory notes with examples can be found in the first 8 pages of [Turhan, 2010] and the first 10 pages of [Sattler, 2007]. We close with describing and illustrating the standard reasoning services for DLs in Section 3.3. Note that DLs and its reasoning services return in Chapter 4 about OWL 2, building upon these foundations.

3.1 DL primer

Description logics (DLs) are a family of knowledge representation languages that are widely used in ontological modelling. An important practical reason for this is that they provide one of the main underpinnings for the Web Ontology Language OWL as standardised by the World Wide Web Consortium (W3C). However, DLs have been used in knowledge representation long before the advent of ontological modelling in the context of the Semantic Web, tracing back to first DL modelling languages in the mid 1980s.

As their name suggests, DLs are logics (in fact they are decidable fragments of first-order logic), and as such they are equipped with a *formal semantics*: a precise specification of the meaning of DL ontologies. This formal semantics allows humans and computer systems to exchange DL ontologies without ambiguity as to their intended meaning, and also makes it possible to use logical deduction to *infer* additional information from the facts stated explicitly in an ontology – an important feature that distinguishes DLs from other modelling languages such as UML.

The capability of inferring additional knowledge increases the modelling power of DLs but it also requires some understanding on the side of the modeller and, above all, good tool support for computing the conclusions. The computation of inferences is called *reasoning* and an important goal of DL language design has been to ensure that reasoning algorithms of good performance are available. This is one of the reasons why there is not just a single description logic: the best balance between expressivity of the language and complexity of reasoning depends on the intended application.

In this paper we provide a self-contained first introduction to description logics. We start by explaining the basic way in which knowledge is modelled in DLs in Section 3.1.1 and continue with an intuitive introduction to the most important DL modelling features in Section 3.1.2. This leads us to the rather expressive DL called $\mathcal{SROIQ}$, the syntax of which we summarise in Section 3.1.3. In Section 3.1.4, we explain the underlying ideas of DL semantics and use it to define the meaning of $\mathcal{SROIQ}$ ontologies. Many DLs can be obtained by omitting some features of $\mathcal{SROIQ}$ and in Section 3.1.5 we review some of the most important DLs obtained in this way. In particular, this includes various light-weight description logics that allow for particularly efficient reasoning.

3.1.1 Basic Building Blocks of DL Ontologies

Description logics (DLs) provide means to model the relationships between entities in a domain of interest. In DLs there are three kinds of entities: concepts, roles and individual names.[1] Concepts denote sets of individuals, roles denote binary relations between the individuals, and individual names denote single individuals in the domain. Readers familiar with first-order logic will recognise these as unary predicates, binary predicates and constants.

[1]In OWL concepts and roles are respectively known as classes and properties; see Chapter 4.

For example, an ontology modelling the domain of people and their family relationships might use concepts such Parent to denote the set of all parents and Female to represent the set of all female individuals, roles such as $\mathsf{parentOf}$ to denote the (binary) relationship between parents and their children, and individual names such as julia and john to denote the individuals Julia and John.

Unlike a database, a DL ontology does not fully describe a particular situation or "state of the world"; rather it consists of a set of statements, called axioms, each of which must be true in the situation described. These axioms typically capture only partial knowledge about the situation that the ontology is describing, and there may be many different states of the world that are consistent with the ontology. Although, from the point of view of logic, there is no principal difference between different types of axioms, it is customary to separate them into three groups: assertional (ABox) axioms, terminological (TBox) axioms and relational (RBox) axioms.

Asserting Facts with ABox Axioms ABox axioms capture knowledge about named individuals, i.e., the concepts to which they belong and how they are related to each other. The most common ABox axioms are *concept assertions* such as

$$\mathsf{Mother}(\mathsf{julia}), \tag{3.1}$$

which asserts that Julia is a mother or, more precisely, that the individual named julia is an *instance* of the concept Mother.

Role assertions describe relations between named individuals. The assertion

$$\mathsf{parentOf}(\mathsf{julia}, \mathsf{john}), \tag{3.2}$$

for example, states that Julia is a parent of John or, more precisely, that the individual named julia is in the relation that is denoted by $\mathsf{parentOf}$ to the individual named john. The previous sentence shows that it can be rather cumbersome to explicitly point out that the relationships expressed by an axiom are really relationships between the individuals, sets and relations that are denoted by the respective individual names, concepts and roles. Assuming that this subtle distinction between syntactic identifiers and semantic entities is understood, we will thus often adopt a more sloppy and readable formulation. Section 3.1.4 below explains the underlying semantics with greater precision.

Although it is intuitively clear that Julia and John are different individuals, this fact does not logically follow from what we have stated so far. DLs do not make the *unique name assumption*, so different names might refer to the same individual unless explicitly stated otherwise. The *individual inequality* assertion

$$\mathsf{julia} \not\approx \mathsf{john} \tag{3.3}$$

is used to assert that Julia and John are actually different individuals. On the other hand, an *individual equality* assertion, such as

$$\mathsf{john} \approx \mathsf{johnny}, \tag{3.4}$$

states that two different names are known to refer to the same individual. Such situations can arise, for example, when combining knowledge about the same domain from several different sources, a task that is known as *ontology alignment*.

Expressing Terminological Knowledge with TBox Axioms TBox axioms describe relationships between concepts. For example, the fact that all mothers are parents is expressed by the *concept inclusion*

$$\mathsf{Mother} \sqsubseteq \mathsf{Parent}, \tag{3.5}$$

in which case we say that the concept Mother is *subsumed* by the concept Parent. Such knowledge can be used to infer further facts about individuals. For example, (3.1) and (3.5) together imply that Julia is a parent.

Concept equivalence asserts that two concepts have the same instances, as in

$$\mathsf{Person} \equiv \mathsf{Human}. \tag{3.6}$$

While synonyms are an obvious example of equivalent concepts, in practice one more often uses concept equivalence to give a name to complex expressions as introduced in Section 3.1.2 below. Furthermore, such additional concept expressions can be combined with equivalence and inclusion to describe more complex situations such as the disjointness of concepts, which asserts that two concepts do not share any instances.

Modelling Relationships between Roles with RBox Axioms RBox axioms refer to properties of roles. As for concepts, DLs support *role inclusion* and *role equivalence* axioms. For example, the inclusion

$$\mathsf{parentOf} \sqsubseteq \mathsf{ancestorOf} \tag{3.7}$$

states that parentOf is a *subrole* of ancestorOf, i.e., every pair of individuals related by parentOf is also related by ancestorOf. Thus (3.2) and (3.7) together imply that Julia is an ancestor of John.

In role inclusion axioms, *role composition* can be used to describe roles such as uncleOf. Intuitively, if Charles is a brother of Julia and Julia is a parent of John, then Charles is an uncle of John. This kind of relationship between the roles brotherOf, parentOf and uncleOf is captured by the *complex role inclusion* axiom

$$\mathsf{brotherOf} \circ \mathsf{parentOf} \sqsubseteq \mathsf{uncleOf}. \tag{3.8}$$

Note that role composition can only appear on the left-hand side of complex role inclusions. Furthermore, in order to retain decidability of reasoning (see the end of Section 3.1.4 for a discussion on decidability), their use is restricted by additional structural restrictions that specify whether or not a collection of such axioms can be used together in one ontology.

Nobody can be both a parent and a child of the same individual, so the two roles parentOf and childOf are disjoint. In DLs we can write *disjoint roles* as follows:

$$\mathit{Disjoint}(\mathsf{parentOf}, \mathsf{childOf}). \tag{3.9}$$

Further RBox axioms include *role characteristics* such as reflexivity, symmetry and transitivity of roles. These are closely related to a number of other DL features and we will discuss them again in more detail in Section 3.1.2.

3.1.2 Constructors for Concepts and Roles

The basic types of axioms introduced in Section 3.1.1 are rather limited for accurate modelling. To describe more complex situations, DLs allow new concepts and roles to be built using a variety of different constructors. We distinguish concept and role constructors depending on whether concept or role expressions are constructed. In the case of concepts, one can further separate basic Boolean constructors, role restrictions and nominals/enumerations. At the end of this section, we revisit the additional kinds of RBox axioms that have been omitted in Section 3.1.1.

Boolean Concept Constructors Boolean concept constructors provide basic Boolean operations that are closely related to the familiar operations of intersection, union and complement of sets, or to conjunction, disjunction and negation of logical expressions.

For example, concept inclusions allow us to state that all mothers are female and that all mothers are parents, but what we really mean is that mothers are *exactly* the female parents. DLs support such statements by allowing us to form complex concepts such as the *intersection* (also called *conjunction*)

$$\mathsf{Female} \sqcap \mathsf{Parent}, \tag{3.10}$$

which denotes the set of individuals that are both female and parents. A complex concept can be used in axioms in exactly the same way as an atomic concept, e.g., in the equivalence $\mathsf{Mother} \equiv \mathsf{Female} \sqcap \mathsf{Parent}$.

Union (also called *disjunction*) is the dual of intersection. For example, the concept

$$\mathsf{Father} \sqcup \mathsf{Mother} \tag{3.11}$$

describes those individuals that are either fathers or mothers. Again, it can be used in an axiom such as $\mathsf{Parent} \equiv \mathsf{Father} \sqcup \mathsf{Mother}$, which states that a parent is either a father or a mother (and vice versa).

Sometimes we are interested in individuals that do *not* belong to a certain concept, e.g., in women who are not married. These could be described by the complex concept

$$\mathsf{Female} \sqcap \neg\mathsf{Married}, \tag{3.12}$$

where the *complement* (also called *negation*) $\neg\mathsf{Married}$ denotes the set of all individuals that are not married.

It is sometimes useful to be able to make a statement about every individual, e.g., to say that everybody is either male or female. This can be accomplished by the axiom

$$\top \sqsubseteq \mathsf{Male} \sqcup \mathsf{Female}, \tag{3.13}$$

where the *top concept* $\top$ is a special concept with every individual as an instance; it can be viewed as an abbreviation for $C \sqcup \neg C$ for an arbitrary concept C. Note that this modelling is rather coarse as it presupposes that every individual has a gender, which may not be reasonable for instances of a concept such as $\mathsf{Computer}$. We will see more useful applications for $\top$ later on.

To express that, for the purposes of our modelling, nobody can be both a male and a female at the same time, we can declare the set of male and the set of female individuals to be disjoint. While ontology languages like OWL provide a basic constructor for disjointness, it is naturally captured in DLs with the axiom

$$\mathsf{Male} \sqcap \mathsf{Female} \sqsubseteq \bot, \tag{3.14}$$

where the *bottom concept* $\bot$ is the dual of $\top$, that is the special concept with no individuals as instances; it can be seen as an abbreviation for $C \sqcap \neg C$ for an arbitrary concept C. The above axiom thus says that the intersection of the two concepts is empty.

Role Restrictions So far we have seen how to use TBox and RBox axioms to express relationships between concepts and roles, respectively. The most interesting feature of DLs, however, is their ability to form statements that link concepts and roles together. For example, there is an obvious relationship between the concept Parent and the role parentOf, namely, a parent is someone who is a parent of at least one individual. In DLs, this relationship can be captured by the concept equivalence

$$\mathsf{Parent} \equiv \exists \mathsf{parentOf}.\top, \tag{3.15}$$

where the *existential restriction* $\exists\mathsf{parentOf}.\top$ is a complex concept that describes the set of individuals that are parents of at least one individual (instance of $\top$). Similarly, the concept $\exists\mathsf{parentOf}.\mathsf{Female}$ describes those individuals that are parents of at least one female individual, i.e., those that have a daughter.

To denote the set of individuals all of whose children are female, we use the *universal restriction*

$$\forall \mathsf{parentOf}.\mathsf{Female}. \tag{3.16}$$

It is a common error to forget that (3.16) also includes those that have no children at all. More accurately (and less naturally), the axiom can be said to describe the set of all individuals that have "no children other than female ones," i.e., no "no children that are not female." Following this wording, the concept (3.16) could indeed be equivalently expressed as $\neg\exists\mathsf{parentOf}.\neg\mathsf{Female}$. If this meaning is not intended, one can describe the individuals who have at least one child and with all their children being female by the concept $(\exists\mathsf{parentOf}.\top) \sqcap (\forall\mathsf{parentOf}.\mathsf{Female})$.

Existential and universal restrictions are useful in combination with the top concept for expressing *domain* and *range restrictions* on roles; that is, restrictions on the kinds of individual that can be in the domain and range of a given role. To restrict the domain of sonOf to male individuals we can use the axiom

$$\exists \mathsf{sonOf}.\top \sqsubseteq \mathsf{Male}, \tag{3.17}$$

and to restrict its range to parents we can write

$$\top \sqsubseteq \forall \mathsf{sonOf}.\mathsf{Parent}. \tag{3.18}$$

In combination with the assertion $\mathsf{sonOf}(\mathsf{john}, \mathsf{julia})$, these axioms would then allow us to deduce that John is male and Julia is a parent. Note how this contrasts with the meaning of *constraints* in databases, which would also allow us to state, e.g., that all sons must be male. However, given only the fact that John is the son of Julia, such a constraint would simply be violated (leading to an error) rather than implying that John is male. Mistaking DL axioms for constraints is a very common source of modelling errors.

Number restrictions allow us to restrict the number of individuals that can be reached via a given role. For example, we can form the *at-least restriction*

$$\geqslant 2\, \mathsf{childOf}.\mathsf{Parent} \tag{3.19}$$

to describe the set of individuals that are children of at least two parents, and the *at-most restriction*

$$\leqslant 2\, \mathsf{childOf}.\mathsf{Parent} \tag{3.20}$$

for those that are children of at most two parents. The axiom $\mathsf{Person} \sqsubseteq \geqslant 2\, \mathsf{childOf}.\mathsf{Parent} \sqcap \leqslant 2\, \mathsf{childOf}.\mathsf{Parent}$ then states that every person is a child of exactly two parents.

Finally, *local reflexivity* can be used to describe the set of individuals that are related to themselves via a given role. For example, the set of individuals that are talking to themselves is described by the concept

$$\exists \mathsf{talksTo}.\mathit{Self}. \tag{3.21}$$

Nominals As well as defining concepts in terms of other concepts (and roles), it may also be useful to define a concept by simply enumerating its instances. For example, we might define the concept Beatle by enumerating its instances: john, paul, george, and ringo. Enumerations are not supported natively in DLs, but they can be simulated in DLs using *nominals.* A nominal is a concept that has exactly one instance. For example, $\{\mathsf{john}\}$ is the concept whose only instance is (the individual denoted by) john. Combining nominals with union, the enumeration in our example could be expressed as

$$\mathsf{Beatle} \equiv \{\mathsf{john}\} \sqcup \{\mathsf{paul}\} \sqcup \{\mathsf{george}\} \sqcup \{\mathsf{ringo}\}. \tag{3.22}$$

It is interesting to note that, using nominals, a concept assertion $\mathsf{Mother}(\mathsf{julia})$ can be turned into a concept inclusion $\{\mathsf{julia}\} \sqsubseteq \mathsf{Mother}$ and a role assertion $\mathsf{parentOf}(\mathsf{julia}, \mathsf{john})$ into a concept inclusion $\{\mathsf{julia}\} \sqsubseteq \exists \mathsf{parentOf}.\{\mathsf{john}\}$. This illustrates that the distinction between ABox and TBox does not have a deeper logical meaning.

Role Constructors In contrast to the variety of concept constructors, DLs provide only few constructor for forming complex roles. In practice, *inverse roles* are the most important such constructor. Intuitively, the relationship between the roles parentOf and childOf is that, for example, if Julia is a parent of John, then John is a child of Julia and vice versa. More formally, parenfOf is the inverse of childOf, which in DLs can be expressed by the equivalence

$$\mathsf{parentOf} \equiv \mathsf{childOf}^{-}, \tag{3.23}$$

where the complex role $\mathsf{childOf}^{-}$ denotes the inverse of childOf.

In analogy to the top concept, DLs also provide the *universal role*, denoted by U, which always relates all pairs of individuals. It typically plays a minor role in modelling,[2] but it establishes symmetry between roles and concepts w.r.t. a top element. Similarly, an *empty role* that corresponds to the bottom concept is also available in OWL but has rarely been introduced as a constructor in DLs; however, we can define any role R to be empty using the axiom $\top \sqsubseteq \neg\exists R.\top$ ("all things do not relate to anything through R"). Interestingly, the universal role cannot be defined by TBox axioms using the constructors introduced above, and in particular universal role restrictions cannot express that a role is universal.

More RBox Axioms: Role Characteristics In Section 3.1.1 we introduced three forms of RBox axioms: role inclusions, role equivalences and role disjointness. OWL provides a variety of others, namely role transitivity, symmetry, asymmetry, reflexivity and irreflexivity. These are sometimes considered as basic axiom types in DLs as well, using some suggestive notation such as $\mathit{Trans}(\mathsf{ancestorOf})$ to express that the role ancestorOf is transitive. However, such axioms are just syntactic sugar; all role characteristics can be expressed using the features of DLs that we have already introduced.

[2]Although there are a few interesting things that could be expressed with U, such as *concept products* [Rudolph et al., 2008a], tool support is rarely sufficient for using this feature in practice.

Transitivity is a special form of complex role inclusion. For example, transitivity of $\mathsf{ancestorOf}$ can be captured by the axiom $\mathsf{ancestorOf} \circ \mathsf{ancestorOf} \sqsubseteq \mathsf{ancestorOf}$. A role is *symmetric* if it is equivalent to its own inverse, e.g., $\mathsf{marriedTo} \equiv \mathsf{marriedTo}^-$, and it is *asymmetric* if it is disjoint from its own inverse, as in $\mathit{Disjoint}(\mathsf{parentOf}, \mathsf{parentOf}^-)$. If desired, *global reflexivity* can be expressed by imposing local reflexivity on the top concept as in $\top \sqsubseteq \exists\mathsf{knows}.\mathit{Self}$. A role is *irreflexive* if it is never locally reflexive, as in the case of $\top \sqsubseteq \neg\exists\mathsf{marriedTo}.\mathit{Self}$.

3.1.3 The Description Logic $\mathcal{SROIQ}$

In this section, we summarise the various features that have been introduced informally above to provide a comprehensive definition of DL syntax. Doing so yields the description logic called $\mathcal{SROIQ}$, which is one of the most expressive DLs commonly considered today. It also largely agrees in expressivity with the ontology language OWL 2 DL, though there are still some differences as explained in Section 4.

Formally, every DL ontology is based on three finite sets of signature symbols: a set N_I of *individual names*, a set N_C of *concept names* and a set N_R of *role names*. Usually these sets are assumed to be fixed for some application and are therefore not mentioned explicitly. Now the set of $\mathcal{SROIQ}$ *role expressions* $\mathbf{R}$ (over this signature) is defined by the following grammar:

$$\mathbf{R} ::= U \mid \mathsf{N}_R \mid \mathsf{N}_R^-$$

where U is the universal role (Section 3.1.2). Based on this, the set of $\mathcal{SROIQ}$ *concept expressions* $\mathbf{C}$ is defined as:

$$\mathbf{C} ::= \mathsf{N}_C \mid (\mathbf{C}\sqcap\mathbf{C}) \mid (\mathbf{C}\sqcup\mathbf{C}) \mid \neg\mathbf{C} \mid \top \mid \bot \mid \exists\mathbf{R}.\mathbf{C} \mid \forall\mathbf{R}.\mathbf{C} \mid \geqslant n\,\mathbf{R}.\mathbf{C} \mid \leqslant n\,\mathbf{R}.\mathbf{C} \mid \exists\mathbf{R}.\mathit{Self} \mid \{\mathsf{N}_I\}$$

where n is a non-negative integer. As usual, expressions like $(\mathbf{C} \sqcap \mathbf{C})$ represent any expression of the form $(C \sqcap D)$ with $C, D \in \mathbf{C}$. It is common to omit parentheses if this cannot lead to confusion with expressions of different semantics. For example, parentheses do not matter for $A \sqcup B \sqcup C$ whereas the expressions $A \sqcap B \sqcup C$ and $\exists R.A \sqcap B$ are ambiguous.

Using the above sets of individual names, roles and concepts, the *axioms* of $\mathcal{SROIQ}$ can be defined to be of the following basic forms:

ABox:	$\mathbf{C}(\mathsf{N}_I)$	$\mathbf{R}(\mathsf{N}_I, \mathsf{N}_I)$	$\mathsf{N}_I \approx \mathsf{N}_I$	$\mathsf{N}_I \not\approx \mathsf{N}_I$
TBox:	$\mathbf{C} \sqsubseteq \mathbf{C}$	$\mathbf{C} \equiv \mathbf{C}$		
RBox:	$\mathbf{R} \sqsubseteq \mathbf{R}$	$\mathbf{R} \equiv \mathbf{R}$	$\mathbf{R} \circ \mathbf{R} \sqsubseteq \mathbf{R}$	$\mathit{Disjoint}(\mathbf{R}, \mathbf{R})$

with the intuitive meanings as explained in Section 3.1.1 and 3.1.2.

Roughly speaking, a $\mathcal{SROIQ}$ ontology (or *knowledge base*) is simply a set of such axioms. To ensure the existence of reasoning algorithms that are correct and terminating, however, additional syntactic restrictions must be imposed on ontologies. These restrictions refer not to single axioms but to the structure of the ontology as a whole, hence they are called *structural restrictions*. The two such conditions relevant for $\mathcal{SROIQ}$ are based on the notions of *simplicity* and *regularity*. Notably, both are automatically satisfied for ontologies that do not contain complex role inclusion axioms.

A role R in an ontology $\mathcal{O}$ is called *non-simple* if some complex role inclusion axiom (i.e., one that uses role composition $\circ$) in $\mathcal{O}$ implies instances of R; otherwise it is called *simple*. A more precise definition of the non-simple role expressions of the ontology $\mathcal{O}$ is given by the following rules:

- if $\mathcal{O}$ contains an axiom $S \circ T \sqsubseteq R$, then R is non-simple,
- if R is non-simple, then its inverse R^- is also non-simple,[3]
- if R is non-simple and $\mathcal{O}$ contains any of the axioms $R \sqsubseteq S$, $S \equiv R$ or $R \equiv S$, then S is also non-simple.

All other roles are called simple.[4] Now for a $\mathcal{SROIQ}$ ontology it is required that the following axioms and concepts contain simple roles only:

$$\begin{array}{rl} \text{Restricted axioms:} & \mathit{Disjoint}(\mathbf{R},\mathbf{R}) \\ \text{Restricted concept expressions:} & \exists\mathbf{R}.\mathit{Self} \qquad \geqslant n\,\mathbf{R}.\mathbf{C} \qquad \leqslant n\,\mathbf{R}.\mathbf{C}. \end{array}$$

The other structural restriction that is relevant for $\mathcal{SROIQ}$ is called *regularity* and is concerned with RBox axioms only. Roughly speaking, the restriction ensures that cyclic dependencies between complex role inclusion axioms occur only in a limited form. For details, please see the pointers given in Section **??**. For the introductory treatment in this paper, it suffices to note that regularity, just like simplicity, is a property of the ontology as a whole that cannot be checked for each axiom individually. An important practical consequence is that the union of two regular ontologies may no longer be regular. This must be taken into account when merging ontologies in practice.

3.1.4 Description Logic Semantics

The formal meaning of DL axioms is given by their semantics. In particular, the semantics specifies what the logical consequences of an ontology are. The formal semantics is therefore the main guideline for every tool that computes logical consequences of DL ontologies, and a basic understanding of its working is vital to make reasonable modelling choices and to comprehend the results given by software applications. Luckily, the semantics of description logics is not difficult to understand provided that some common misconceptions are avoided.

Intuitively speaking, an ontology describes a particular situation in a given domain of discourse. For example, the axioms in Sections 3.1.1 and 3.1.2 describe a particular situation in the "families and relationships" domain. However, ontologies usually cannot fully specify the situation that they describe. On the one hand, there is no formal relationship between the symbols we use and the objects that they represent: the individual name julia, for example, is just a syntactic identifier with no intrinsic meaning. Indeed, the intended meaning of the identifiers in our ontologies has no influence on their formal semantics: what we know about them stems only from the ontological axioms. On the other hand, the axioms in an ontology typically do not provide complete information. For example, (3.3) and (3.4) in Section 3.1.1 state that some individuals are equal and that others are unequal, but in many other cases this information might be left unspecified.

Description logics have been designed to deal with such incomplete information. Rather than making default assumptions in order to fully specify one particular interpretation for each ontology, the DL semantics generally considers all the possible situations (i.e., states of the world) where the axioms of an ontology would hold (we also say: where the axioms are *satisfied*). This characteristic is sometimes called the *Open World*

[3]If $R = S^-$ already is an inverse role, then R^- should be read as S. We do not allow expressions like S^{--}.

[4]Whether the universal role U is simple or not is a matter of preference that does not affect the computational properties of the logic [Rudolph et al., 2008b]. However, the universal role in OWL 2 is considered non-simple.

Table 3.1: Syntax and semantics of $\mathcal{SROIQ}$ constructors.

	Syntax	*Semantics*
Individuals:		
individual name	a	$a^{\mathcal{I}}$
Roles:		
atomic role	R	$R^{\mathcal{I}}$
inverse role	R^{-}	$\{\langle x,y\rangle \mid \langle y,x\rangle \in R^{\mathcal{I}}\}$
universal role	U	$\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
Concepts:		
atomic concept	A	$A^{\mathcal{I}}$
intersection	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
union	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
complement	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
top concept	$\top$	$\Delta^{\mathcal{I}}$
bottom concept	$\bot$	$\emptyset$
existential restriction	$\exists R.C$	$\{x \mid$ some $R^{\mathcal{I}}$-successor of x is in $C^{\mathcal{I}}\}$
universal restriction	$\forall R.C$	$\{x \mid$ all $R^{\mathcal{I}}$-successors of x are in $C^{\mathcal{I}}\}$
at-least restriction	$\geqslant n\, R.C$	$\{x \mid$ at least n $R^{\mathcal{I}}$-successors of x are in $C^{\mathcal{I}}\}$
at-most restriction	$\leqslant n\, R.C$	$\{x \mid$ at most n $R^{\mathcal{I}}$-successors of x are in $C^{\mathcal{I}}\}$
local reflexivity	$\exists R.\mathit{Self}$	$\{x \mid \langle x,x\rangle \in R^{\mathcal{I}}\}$
nominal	$\{a\}$	$\{a^{\mathcal{I}}\}$

where $a, b \in \mathsf{N}_I$ are individual names, $A \in \mathsf{N}_C$ is a concept name, $C, D \in \mathbf{C}$ are concepts, and $R \in \mathbf{R}$ is a role

Assumption since it keeps unspecified information open.[5] A logical consequence of an ontology is an axiom that holds in all interpretations that satisfy the ontology, i.e., something that is true in all conceivable states of the world that agree with what is said in the ontology. The more axioms an ontology contains, the more specific are the constraints that it imposes on possible interpretations, and the fewer interpretations exist that satisfy all of the axioms. Conversely, if fewer interpretations satisfy an ontology, then more axioms hold in all of them, and more logical consequences follow from the ontology. The previous two sentences imply that the semantics of description logics is *monotonic*: additional axioms always lead to additional consequences, or, more informally, the more knowledge we feed into a DL system the more results it returns.

An extreme case is when an ontology is not satisfied in any interpretation. The ontology is then called *unsatisfiable* or *inconsistent*. In this case *every* axiom holds vacuously in all of the (zero) interpretations that satisfy the ontology. Such an ontology is clearly of no utility, and avoiding inconsistency (and checking for it in the first place) is therefore an important task during modelling.

We have outlined above the most important ideas of DL semantics. What remains to be done is to define what we really mean by an "interpretation" and which conditions must hold for particular axioms to be satisfied by an interpretation. For this, we closely follow the intuitive ideas established above: an interpretation $\mathcal{I}$ consists of a set $\Delta^{\mathcal{I}}$ called the *domain* of $\mathcal{I}$ and an interpretation function $\cdot^{\mathcal{I}}$ that maps each atomic concept A to a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, each atomic role R to a binary relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, and each

[5]A *Closed World Assumption* "closes" the interpretation by assuming that every fact not explicitly stated to be true is actually false. Both terms are not formally specified and rather outline the general flavour of a semantics than any particular definition.

Table 3.2: Syntax and semantics of $\mathcal{SROIQ}$ axioms.

	Syntax	*Semantics*
ABox:		
concept assertion	$C(a)$	$a^{\mathcal{I}} \in C^{\mathcal{I}}$
role assertion	$R(a,b)$	$\langle a^{\mathcal{I}}, b^{\mathcal{I}} \rangle \in R^{\mathcal{I}}$
individual equality	$a \approx b$	$a^{\mathcal{I}} = b^{\mathcal{I}}$
individual inequality	$a \not\approx b$	$a^{\mathcal{I}} \neq b^{\mathcal{I}}$
TBox:		
concept inclusion	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
concept equivalence	$C \equiv D$	$C^{\mathcal{I}} = D^{\mathcal{I}}$
RBox:		
role inclusion	$R \sqsubseteq S$	$R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$
role equivalence	$R \equiv S$	$R^{\mathcal{I}} = S^{\mathcal{I}}$
complex role inclusion	$R_1 \circ R_2 \sqsubseteq S$	$R_1^{\mathcal{I}} \circ R_2^{\mathcal{I}} \subseteq S^{\mathcal{I}}$
role disjointness	$\mathit{Disjoint}(R,S)$	$R^{\mathcal{I}} \cap S^{\mathcal{I}} = \emptyset$

individual name a to an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. The interpretation of complex concepts and roles follows from the interpretation of the basic entities. Table 3.1 shows how to obtain the semantics of each compound expression from the semantics of its parts. By "$R^{\mathcal{I}}$-successor of x" we mean any individual y such that $\langle x, y \rangle \in R^{\mathcal{I}}$. The definition should confirm the intuitive explanations given for each case in Section 3.1.2. For example, the semantics of $\mathsf{Female} \sqcap \mathsf{Parent}$ is indeed the intersection of the semantics of Female and Parent.

Since an interpretation $\mathcal{I}$ fixes the meaning of all entities, we can unambiguously say for each axiom whether it holds in $\mathcal{I}$ or not. An axiom *holds* in $\mathcal{I}$ (we also say $\mathcal{I}$ *satisfies* α and write $\mathcal{I} \models \alpha$) if the corresponding condition in Table 3.2 is met. Again, these definitions fully agree with the intuitive explanations given in Section 3.1.1. If all axioms in an ontology $\mathcal{O}$ hold in $\mathcal{I}$ (i.e., if $\mathcal{I}$ satisfies $\mathcal{O}$, written $\mathcal{I} \models \mathcal{O}$), then $\mathcal{I}$ is a *model* of $\mathcal{O}$. Thus a model is an abstraction of a state of the world that satisfies all axioms in the ontology. An ontology is *consistent* if it has at least one model. An axiom α is a *consequence* of an ontology $\mathcal{O}$ (or $\mathcal{O}$ *entails* α written $\mathcal{O} \models \alpha$) if α holds in every model of $\mathcal{O}$. In particular, an inconsistent ontology entails every axiom.

A noteworthy consequence of this semantics is the meaning of individual names in DL ontologies. We already remarked that DLs do not usually make the Unique Name Assumption, and indeed our formal definition allows two individual names to be interpreted as the same individual (element of the domain). Possibly even more important is the fact that the domain of an interpretation is allowed to contain many individuals that are not denoted by any individual name. A common confusion in modelling arises from the implicit assumption that interpretations must only contain individuals that are denoted by individual names (such individuals are also called *named individuals*). For example, one could wrongly assume the ontology consisting of the axioms

$$\mathsf{parentOf}(\mathsf{julia}, \mathsf{john}) \qquad \mathsf{manyChildren}(\mathsf{julia}) \qquad \mathsf{manyChildren} \sqsubseteq {\geqslant}3\, \mathsf{parentOf}.\top$$

to be inconsistent since it requires Julia to have at least 3 children when only one (John) is given. However, there are many conceivable models where Julia does have three children, even though only one of them is explicitly named. A significant number of modelling errors can be traced back to similar misconceptions that are easy to prevent if the general open world assumption of DLs is kept in mind.

Another point to note is that the above specification of the semantics does not provide any hint as to how to compute the relevant entailments in practical software tools. There are infinitely many possible interpretations, each of which may have an infinite domain (in fact there are some ontologies that are satisfied only by interpretations with infinite domains). Therefore it is impossible to test all interpretations to see if they model a given ontology, and impossible to test all models of an ontology to see if they entail a given axiom. Rather, one has to devise concrete deduction procedures and prove their correctness with respect to the above specification. The interplay of certain expressive features can make reasoning algorithms more complicated and in some cases it can even be shown that no correct and terminating algorithm exists at all (i.e., that reasoning is undecidable). For our purposes it suffices to know that entailment of axioms is decidable for $\mathcal{SROIQ}$ (with the structural restrictions explained in Section 3.1.3) and that a number of free and commercial tools are available. Such tools are typically optimised for more specific reasoning problems, such as consistency checking, the entailment of concept subsumptions (subsumption checking) or of concept assertions (instance checking). Many of these standard inferencing problems can be expressed in terms of each other, so they can be handled by very similar reasoning algorithms.

3.1.5 Important Fragments of $\mathcal{SROIQ}$

Many different description logics have been introduced in the literature. Typically, they can be characterised by the types of constructors and axioms that they allow, which are often a subset of the constructors in $\mathcal{SROIQ}$. For example, the description logic $\mathcal{ALC}$ is the fragment of $\mathcal{SROIQ}$ that allows no RBox axioms and only $\sqcap$, $\sqcup$, $\neg$, $\exists$ and $\forall$ as its concept constructors. It is often considered the most basic DL. The extension of $\mathcal{ALC}$ with transitive roles is traditionally denoted by the letter $\mathcal{S}$. Some other letters used in DL names hint at a particular constructor, such as inverse roles $\mathcal{I}$, nominals $\mathcal{O}$, qualified number restrictions $\mathcal{Q}$, and role hierarchies (role inclusion axioms without composition) $\mathcal{H}$. So, for example, the DL named $\mathcal{ALCHIQ}$ extends $\mathcal{ALC}$ with role hierarchies, inverse roles and qualified number restrictions. The letter $\mathcal{R}$ most commonly refers to the presence of role inclusions, local reflexivity *Self*, and the universal role U, as well as the additional role characteristics of transitivity, symmetry, asymmetry, role disjointness, reflexivity, and irreflexivity. This naming scheme explains the name $\mathcal{SROIQ}$.

In recent years, fragments of DLs have been specifically developed in order to obtain favourable computational properties. For this purpose, $\mathcal{ALC}$ is already too large, since it only admits reasoning algorithms that run in worst-case exponential time. More light-weight DLs can be obtained by further restricting expressivity, while at the same time a number of additional $\mathcal{SROIQ}$ features can be added without loosing the good computational properties. The three main approaches for obtaining light-weight DLs are $\mathcal{EL}$, *DLP* and *DL-Lite*, which also correspond to language fragments OWL EL, OWL RL and OWL QL of the Web Ontology Language.

The $\mathcal{EL}$ family of description logics is characterised by allowing unlimited use of existential quantifiers and concept intersection. The original description logic $\mathcal{EL}$ allows only those features and $\top$ but no unions, complements or universal quantifiers, and no RBox axioms. Further extensions of this language are known as $\mathcal{EL}^+$ and $\mathcal{EL}^{++}$. The largest such extension allows the constructors $\sqcap$, $\top$, $\bot$, $\exists$, *Self*, nominals and the universal role, and it supports all types of axioms other than role symmetry, asymmetry and irreflexivity. Interestingly, all standard reasoning tasks for this DL can still be solved in worst-case polynomial time. One can even drop the structural restriction of regularity that is important for $\mathcal{SROIQ}$. $\mathcal{EL}$-type ontologies have been used to model large but light-weight ontologies that consist mainly of terminological data, in particular in the life sciences. A number of reasoners are specifically optimised for handling $\mathcal{EL}$-type ontologies, the most recent of which is the ELK reasoner for OWL EL.[6]

DLP is short for *Description Logic Programs* and comprises various DLs that are syntactically restricted in such a way that axioms could also be read as rules in first-order Horn logic without function symbols. Due to this, DLP-type logics can be considered as kinds of rule languages (hence the name OWL RL) contained in DLs. To accomplish this, one has to allow different syntactic forms for subconcepts and superconcepts in concept inclusion axioms. We do not provide the details here. While DLs in general may require us to consider domain elements that are not denoted by individual names, for DLP one can always restrict attention to models in which all domain elements are denoted by individual names. This is why DLP is often used to augment databases (interpreted as sets of ABox axioms), e.g., in an implementation of OWL RL in the Oracle 11g database management system.

DL-Lite is a family of DLs that is also used in combination with large data collections and existing databases, in particular to augment the expressivity of a query language that retrieves such data. This approach, known as Ontology Based Data Access, considers ontologies as a language for constructing *views* or *mapping rules* on top of existing data. The core feature of DL-Lite is that data access can be realised with standard query languages such as SQL that are not aware of the DL semantics. Ontological information is merely used in a query preprocessing step. Like DLP, DL-Lite requires different syntactic restrictions for subconcepts and superconcepts. We do not present the details here.

3.2 A basic DL: $\mathcal{ALC}$

Section 3.1.5 mentioned that there are "important fragments" of $\mathcal{SROIQ}$, one of them being $\mathcal{ALC}$, which will be introduced now. The two reasons for this are that it is less complicated for illustrating the tableau algorithm for (automated) reasoning later on regarding Exercise 6 about vegetarians and vegans, and to practice a bit with the DL notation in a more condensed notation. The terminology used in this section has been introduced either in the previous section or the previous chapter.

The DL language $\mathcal{ALC}$ ($\mathcal{A}$ttributive $\mathcal{L}$anguage with $\mathcal{C}$oncept negation) contains the following elements:

- *Concepts* denoting entity types/classes/unary predicates/universals, including top $\top$ and bottom $\bot$;
- *Roles* denoting relationships/associations/n-ary predicates/properties;
- *Constructors*: and $\sqcap$, or $\sqcup$, and not $\neg$; quantifiers forall $\forall$ and exists $\exists$
- *Complex concepts* using constructors: Let C and D be concept names, R a role name, then
  – $\neg C$, $C \sqcap D$, and $C \sqcup D$ are concepts, and
  – $\forall R.C$ and $\exists R.C$ are concepts
- *Individuals*

Some examples that can be represented in $\mathcal{ALC}$ are:

- Concepts (primitive, atomic): `Book`, `Course`
- Roles: `ENROLLED`, `READS`

[6] `http://elk-reasoner.googlecode.com/`

- Complex concepts:
  - $\texttt{Student} \sqsubseteq \exists\texttt{ENROLLED}.(\texttt{Course} \sqcup \texttt{DegreeProgramme})$ (this is a *primitive concept*)
  - $\texttt{Mother} \sqsubseteq \texttt{Woman} \sqcap \exists\texttt{PARENTOF}.\texttt{Person}$
  - $\texttt{Parent} \equiv (\texttt{Male} \sqcup \texttt{Female}) \sqcap \exists\texttt{PARENTOF}.\texttt{Mammal} \sqcap \exists\texttt{CARESFOR}.\texttt{Mammal}$ (this is a *defined concept*)
- Individuals in the ABox: $\texttt{Student(Andile)}$, $\texttt{Mother(Katniss)}$, $\neg\texttt{Student(Katniss)}$, $\texttt{ENROLLED(Andile, COMP101)}$

Again, the meaning is defined by the *semantics* of $\mathcal{ALC}$. First, we have a *domain of interpretation*, and an *interpretation* (recollect FOL and model-theoretic semantics from Section 2.1), where:

- Domain Δ is a non-empty set of objects
- Interpretation: $\cdot^{\mathcal{I}}$ is the *interpretation function*, domain $\Delta^{\mathcal{I}}$
  - $\cdot^{\mathcal{I}}$ maps every concept name A to a subset $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
  - $\cdot^{\mathcal{I}}$ maps every role name R to a subset $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
  - $\cdot^{\mathcal{I}}$ maps every individual name a to elements of $\Delta^{\mathcal{I}}$: $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$
- Note: $\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$ and $\bot^{\mathcal{I}} = \emptyset$

Using the typical notation where C and D are concepts, R a role, and a and b are individuals, then they have the following meaning, with on the left-hand side of the "=" the syntax of $\mathcal{ALC}$ under an interpretation and on the right-hand side its semantics:

- $(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \backslash C^{\mathcal{I}}$
- $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$
- $(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$
- $(\forall R.C)^{\mathcal{I}} = \{x \mid \forall y.R^{\mathcal{I}}(x,y) \to C^{\mathcal{I}}(y)\}$
- $(\exists R.C)^{\mathcal{I}} = \{x \mid \exists y.R^{\mathcal{I}}(x,y) \land C^{\mathcal{I}}(y)\}$

Then, we also can specify the notion of *satisfaction*:

- An interpretation $\mathcal{I}$ satisfies the statement $C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
- An interpretation $\mathcal{I}$ satisfies the statement $C \equiv D$ if $C^{\mathcal{I}} = D^{\mathcal{I}}$
- $C(a)$ is satisfied by $\mathcal{I}$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$
- $R(a,b)$ is satisfied by $\mathcal{I}$ if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$
- An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is a *model* of a knowledge base $\mathcal{KB}$ if every axiom of $\mathcal{KB}$ is satisfied by $\mathcal{I}$
- A knowledge base $\mathcal{KB}$ is said to be *satisfiable* if it admits a model

Many DLs have be defined over the past 25 years and their complexity proved. For instance, one could add $\mathcal{I}$nverses to $\mathcal{ALC}$, giving $\mathcal{ALCI}$, or a $\mathcal{H}$ierarchy of roles, $\mathcal{ALCH}$, or $\mathcal{Q}$ualified cardinality restrictions; the appendix of the DL Handbook [Baader et al., 2008] has the full list of letters and the features they denote. You also may like to have a look at the DL Complexity Navigator[7]. In the next chapter about OWL 2 we shall introduce a few more expressive languages, whereas ontology-based data access in Chapter 8 introduces DL-Lite, that is less expressive than $\mathcal{ALC}$ (but more complicated for modelling).

3.3 Reasoning services

The reasoning services for DLs can be divided into so-called 'standard' reasoning services and 'non-standard' reasoning services, where the former are more common and provided

[7] `http://www.cs.man.ac.uk/ezolin/logic/complexity.html`

by all extant DL reasoners, whereas for the latter new problems had been defined, and therewith needing specific algorithms, extensions, and interfaces. We will see an example of the latter in Section 7.4.

3.3.1 Standard reasoning services

The standard reasoning services are as follows (generally, all DL-focussed automated reasoners offer these services).

- Consistency of the knowledge base ($\mathcal{KB} \not\models \top \sqsubseteq \bot$)
  - Is the $\mathcal{KB} = (\mathcal{T}, \mathcal{A})$ consistent (non-selfcontradictory), i.e., is there at least one model for $\mathcal{KB}$, i.e.: can *all* concepts and roles be instantiated without leading to a contradiction?
- Concept (and role) satisfiability ($\mathcal{KB} \not\models C \sqsubseteq \bot$)
  - is there a model of $\mathcal{KB}$ in which C (resp. R) has a nonempty extension, i.e., 'can have instances without leading to contradictions'?
- Concept (and role) subsumption ($\mathcal{KB} \models C \sqsubseteq D$)
  - i.e., is the extension of C (resp. R) contained in the extension of D (resp. S) in every model of $\mathcal{T}$, i.e., 'are all instances of C also instances of D?
- Instance checking ($\mathcal{KB} \models C(a)$ or $\mathcal{KB} \models R(a, b)$)
  - is a (resp. (a, b)) a member of concept C (resp. R) in $\mathcal{KB}$, i.e., is the fact $C(a)$ (resp. $R(a, b)$) satisfied by every interpretation of $\mathcal{KB}$?
- Instance retrieval ($\{a \mid \mathcal{KB} \models C(a)\}$)
  - find all members of C in $\mathcal{KB}$, i.e., compute all individuals a s.t. $C(a)$ is satisfied by every interpretation of $\mathcal{KB}$

You have used the underlying idea of concept subsumption both with EER and UML class diagrams, but then you did it all manually. Now, instead of you having to model a hierarchy of entity types/classes, we let the automated reasoner do it for us thanks to the properties we have represented for the DL concepts.

The following two examples illustrate logical implication and concept subsumption.

Example 3.1. Logical implication Let us first look at logical implication—i.e., $\mathcal{KB} \models \phi$ if every model of $\mathcal{KB}$ is a model of ϕ—with the following example:

- TBox: $\exists\mathsf{TEACHES.Course} \sqsubseteq \neg\mathsf{Undergrad} \sqcup \mathsf{Professor}$
  "The objects that teaches a course are not undergrads or professors"
- ABox: $\mathsf{TEACHES(Thembi, cs101)}$, $\mathsf{Course(cs101)}$, $\mathsf{Undergrad(Thembi)}$

This is depicted graphically in Figure 3.2. What does it entail, if anything? The only possibility to keep this logical theory consistent and satisfiable is to make Thembi a professor, i.e., $\mathcal{KB} \models \mathsf{Professor(Thembi)}$, because anything that teaches a course must be either not an undergrad or a professor. Given that Thembi is an undergrad, she cannot be not an undergrad, hence, she has to be a professor. ♢

What will happen if we have the following knowledge base?

- TBox: $\exists\mathsf{TEACHES.Course} \sqsubseteq \mathsf{Undergrad} \sqcup \mathsf{Professor}$
- ABox: $\mathsf{TEACHES(Thembi, cs101)}$, $\mathsf{Course(cs101)}$, $\mathsf{Undergrad(Thembi)}$

That is, do we obtain $\mathcal{KB} \models \mathsf{Professor(Thembi)}$ again? No.
Perhaps the opposite, that $\mathcal{KB} \models \neg\mathsf{Professor(Thembi)}$? No. Can you explain why?

Example 3.2. Concept subsumption As an example of concept subsumption, consider the following knowledge based $\mathcal{K}$, which is depicted graphically in Figure 3.3:

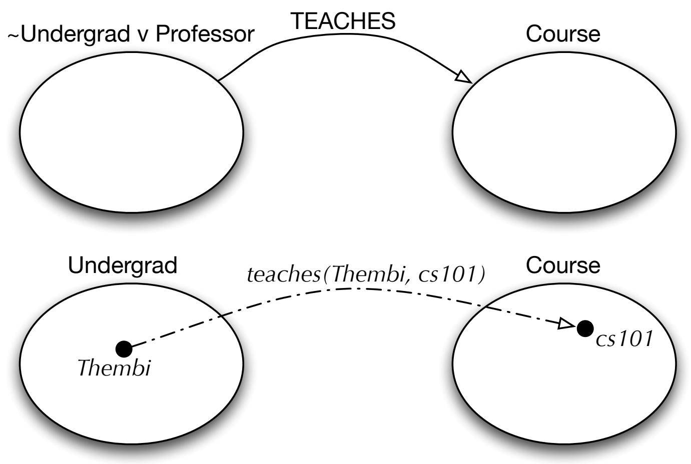


Figure 3.2: Top: Depiction of the TBox according to the given axiom; bottom: depiction of the ABox.

- HonsStudent $\equiv$ Student $\sqcap$ $\exists$ENROLLED.BScHonsDegree
- X $\equiv$ Student $\sqcap$ $\exists$ENROLLED.BScHonsDegree $\sqcap$ $\exists$HASDUTY.TeachingAssistantShip
- Y $\equiv$ Student $\sqcap$ $\exists$ENROLLED.BScHonsDegree $\sqcap$ $\exists$HASDUTY.ProgrammingTask
- X(John), BScHonsDegree(cs12), TeachingAssistantShip(COMP314-W12), ENROLLED(John, cs12), HASDUTY(John, COMP314-W12)

$\mathcal{K} \models$ X $\sqsubseteq$ HonsStudent, i.e.: is the extension of X contained in the extension of HonsStudent in every model of $\mathcal{K}$? Yes. Why? We know that both HonsStudent and X are subclasses of Student and that both are ENROLLED in an BScHonsDegree programme. In addition, every instance of X *also* has a duty performing a TeachingAssistantShip for an undergrad module, whereas, possibly, not all honours students work as a teaching assistant. Thus, all X's are always also an instance of HonsStudent in every possible model of $\mathcal{K}$, hence $\mathcal{K} \models$ X $\sqsubseteq$ HonsStudent. And likewise for $\mathcal{K} \models$ Y $\sqsubseteq$ HonsStudent. This deduction is depicted in green in Figure 3.4. The overall situation is perhaps clearer when we remove all the duplications, as shown in Figure 3.5.

Let us modify this a bit by adding the following two axioms to $\mathcal{K}$:

- Z $\equiv$ Student $\sqcap$ $\exists$ENROLLED.BScHonsDegree $\sqcap$
  $\exists$HASDUTY.(ProgrammingTask $\sqcap$ TeachingAssistantShip)
- TeachingAssistantShip $\sqsubseteq$ $\neg$ProgrammingTask

What happens now? The first step is to look at Z: it has the same properties as HonsStudent, X, and Y, but now we see that each instance of Z has as duty both a ProgrammingTask and TeachingAssistantShip; hence, it must be a subconcept of both X and Y, because it refines them both. So far, so good. The second axiom tells us that the intersection of ProgrammingTask and TeachingAssistantShip is empty, or: they are disjoint, or: there is no object that is both a teaching assistantship for some module and a programming task. But each instance of Z has as duty to carry out a duty that is both a teaching assistantship and a programming task! This object cannot exist, hence, there cannot be a model where Z is instantiated, hence, Z is an unsatisfiable concept. ♢

3.3.2 Techniques

The description of the deductions illustrated in the previous paragraph is an informal, high-level way of describing what the automated reasoner does when computing the

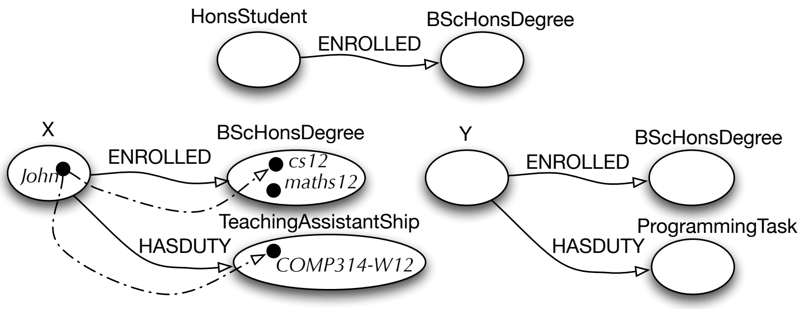


Figure 3.3: Graphical depiction of $\mathcal{K}$ before checking concept subsumption.

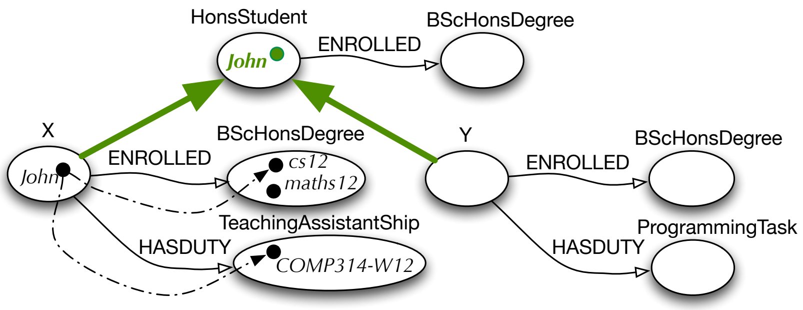


Figure 3.4: Graphical depiction of $\mathcal{K}$ after checking concept subsumption; content in green is deduced.

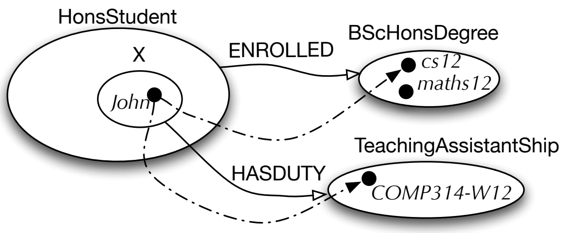


Figure 3.5: Graphical depiction of $\mathcal{K}$ after checking concept subsumption, condensed version and Y not shown.

concept hierarchy and checking for satisfiability. Clearly, such an informal way will not work as an algorithm to be implemented in a computer. There are several proof techniques both in theory and in practice to realise the reasoning service. The most widely used technique at the time of writing (within the scope of DLs and the Semantic web) is *tableau reasoning*, and is quite alike what we have seen with tableau with full FOL. In short, it:

1. Unfolds the TBox
2. Converts the result into negation normal form
3. Applies the tableau rules to generate more Aboxes
4. Stops when none of the rules are applicable

Then:

- $\mathcal{T} \vdash C \sqsubseteq D$ if all Aboxes contain clashes
- $\mathcal{T} \nvdash C \sqsubseteq D$ if some Abox does not contain a clash

First, recall that one enters the tableau in Negation Normal Form (NNF), i.e., "$\neg$" only in front of concepts. For DLs and C and D are concepts, R a role, we use equivalences to obtain NNF:

- $\neg\neg C$ gives C
- $\neg(C \sqcap D)$ gives $\neg C \sqcup \neg D$
- $\neg(C \sqcup D)$ gives $\neg C \sqcap \neg D$
- $\neg(\forall R.C)$ gives $\exists R.\neg C$
- $\neg(\exists R.C)$ gives $\forall R.\neg C$

(This look familiar to you from the logic you had in previous courses.)

Second, there are the tableau rules:
$\sqcap$-rule: If $(C_1 \sqcap C_2)(a) \in S$ but S does not contain both $C_1(a)$ and $C_2(a)$, then
$S = S \cup \{C_1(a), C_2(a)\}$
$\sqcup$-rule: If $(C_1 \sqcup C_2)(a) \in S$ but S contains neither $C_1(a)$ nor $C_2(a)$, then
$S = S \cup \{C_1(a)\}$
$S = S \cup \{C_2(a)\}$
$\forall$-rule: If $(\forall R.C)(a) \in S$ and S contains $R(a, b)$ but not $C(b)$, then
$S = S \cup \{C(b)\}$
$\exists$-rule: If $(\exists R.C)(a) \in S$ and there is no b such that $C(b)$ and $R(a, b)$, then
$S = S \cup \{C(b), R(a, b)\}$

With these ingredients, you can construct a tableau to prove that the aforementioned deductions hold. There will be an exercise about it, and we will see more aspects of automated reasoning in the lectures and exercises about OWL.

3.4 Exercises

Exercise 5. Explain in your own words what the following $\mathcal{ALC}$ reasoning tasks involve and why they are important for reasoning with ontologies:

a. Instance checking.
b. Subsumption checking.
c. Checking for concept satisfiability.

Exercise 6. Consider the following TBox $\mathcal{T}$:
$Vegan \equiv Person \sqcap \forall eats.Plant$
$Vegetarian \equiv Person \sqcap \forall eats.(Plant \sqcup Dairy)$
We want to know if $\mathcal{T} \vdash Vegan \sqsubseteq Vegetarian$.
This we convert to a constraint system $S = \{(Vegan \sqcap \neg Vegetarian)(a)\}$,
which is unfolded (here: complex concepts on the left-hand side are replaced with their properties declared on the right-hand side) into:

$$S = \{Person \sqcap \forall eats.Plant \sqcap \neg(Person \sqcap \forall eats.(Plant \sqcup Dairy))(a)\} \qquad (3.24)$$

Tasks:

a. Rewrite (Eq. 3.24) into negation normal form
b. Enter the tableau by applying the rules (see lecture slides/previous section) until either you find a completion or only clashes.
c. $\mathcal{T} \vdash Vegan \sqsubseteq Vegetarian$?

3.5 Literature and reference material

This material is listed mainly for the curious, not that you have to read it all.

1. Ulrike Sattler. Reasoning in description logics: Basics, extensions, and relatives. In G. Antoniou et al., editors, *Reasoning Web 2007*, volume 4636 of LNCS, page 154182. Springer, 2007. OR Anni-Yasmin Turhan. Reasoning and explanation in EL and in expressive Description Logics. In U. Assmann, A. Bartho, and C. Wende, editors, *Reasoning Web 2010*, volume 6325 of LNCS, pages 127. Springer, 2010.
2. F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider (Eds). *The Description Logics Handbook.* Cambridge University Press, 2009. Chapters 1 and 2.

CHAPTER 4

The Web Ontology Language OWL 2

The FOL and DLs we have seen hasn't gotten us close to implementations. This is going to change in this chapter, where we will look at 'implementation versions' of DLs that have rich tooling support. We will take a look at the computational use of DLs with the so-called *serialization* to obtain computer-processable versions of an ontology and *automated reasoning* over it. The language that we will use to serialise the ontology is the most widely used ontology language for computational purposes, being the Web Ontology Language OWL. It has been standardised first in 2004 and a version 2 in 2009, which has fuelled tool development and deployment of ontologies in ontology-driven information systems. The once thing is that step toward implementation, but it also ends up in a situation where, as mentioned in Section 1.1—what does an ontology look like—, there are various ways of representing the an ontology, and a myriad of syntaxes to cater for a range of preferences and tools.

OWL does not exist in isolation, but is part of the Semantic Web stack—also called the (in)famous 'layer cake'—to make the Semantic Web work. This layer cake is shown in Figure 4.1: at the bottom of the stack, we have XML, which is a surface syntax that has no semantics, and then XML Schema, which describes structure of XML documents. RDF is intended for describing data and facilitating data exchange; it is a datamodel for "relations" between "things", which also has a RDF Schema and an RDF Vocabulary Definition Language. RDF data can be queried with the SPARQL query language (one can draw an analogue with SQL for relational databases, but then tailored to the Internet). On top of that, we have the ontology language for the Web, OWL, to handle the knowledge and reasoning, and rules (RIF). The details of the "trust" and "crypto" are still sketchy, but there are plentiful user interfaces and Semantic Web applications. This chapter focuses on OWL only; however, because it is built upon RDF and XML, we will come across some of that notation as well, as OWL ontologies are serialised (made machine-processable) in those languages.

OWL actually constitutes a family of languages consisting of so-called OWL "species", and we will focus on those species that are based on DLs, which are all fragments of the most expressive one, OWL 2 DL. OWL 2 DL is based on $\mathcal{SROIQ}$ with data types, which we already have come across in Chapter 3. To understand how OWL 2 DL came about in the way it is, with these features and not others, we will touch upon OWL and focus

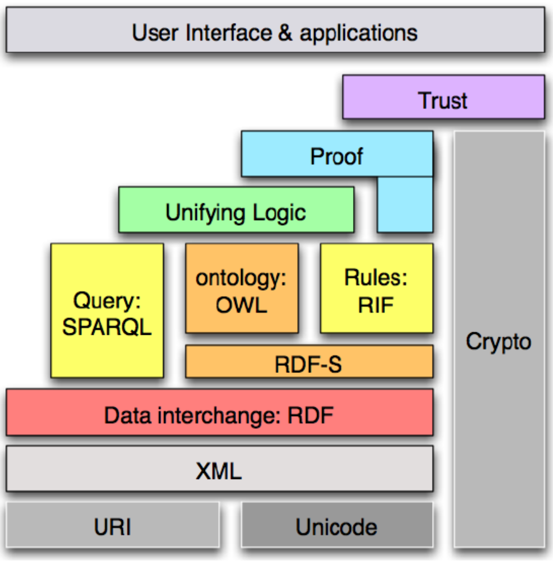


Figure 4.1: The Semantic Web layer cake.

on OWL 2[1].

4.1 Standardizing an ontology language

Before OWL, there were a plethora of ontology languages, such as the `obo` format (directed acyclic graphs) initiated by the Gene Ontology Consortium[2] and is still used widely for bio-ontologies, KL-ONE, and F-logic (frames, and older versions of the Protégé ontology development environment). Unsurprisingly, this caused ontology interoperation problems even at the syntactic level and hampered development and use of ontology tools. To solve those issues, researchers set out to standardise a logic language. RDFS was already in the layer cake of the Semantic Web (Figure 4.1), but appeared to have some problems to function as ontology language—expressiveness limitations, and syntactic and semantic issues [Horrocks et al., 2003]—which is what OWL also aimed to address so as to provide a comprehensive ontology language for the Semantic Web. Further, several predecessors to OWL had a considerable influence on the final product, most notably the SHOE, DAML-ONT, OIL, and DAML+OIL languages, and, more generally, the fruits of 20 years of research on languages and prototyping of automated reasoners by the Description Logics (DL) community. A document of requirements and objectives was devised to specify what such an ontology language for the Semantic Web should meet; the "making of an ontology language" article [Horrocks et al., 2003] gives a general historical view. It also summarises OWL with its three species (OWL lite, OWL-DL, and OWL full) and the details of the standard are freely available[3].

What makes OWL a Semantic Web language compared to the regular DL languages as we have seen in the previous chapter, is that:

– OWL uses URI references as names (like used in RDF); e.g., `http://www.mysite.co.za/UniOnto.owl#Student` is the URI of the class `Student` in the ontology `UniOnto.owl`

[1]There are many learning resources for OWL and OWL2 and therefore Sections 4.1 and 4.2 are kept short. A general non-technical overview is provided in the OWL primer: `http://www.w3.org/TR/owl2-primer/`. The first part of Section 4.3 is also brief; more material is available upon request.

[2]`http://www.geneontology.org`; recall also Figure 1.6.

[3]`http://www.w3.org/TR/owl-ref/`

that is online available from `http://www.mysite.co.za`;

– it gathers information into ontologies stored as documents written in RDF/XML including things like `owl:imports` to import one ontology into another; e.g. another ontology, `HigherEd.owl` can import the `UniOnto.owl` so that the class `http://www.mysite.co.za/UniOnto.owl#Student` becomes, in a way, part of `HigherEd.owl` (though it still exists independently as well);

– it adds RDF data types and XML schema data types for the ranges of data properties (attributes), so one can use, e.g., `string` and `integer` in a similar way as you are familiar with in UML class diagrams and databases.

OWL and OWL 2 will be summarized in the lecture; the following subsections are, roughly, a fragment of the lecture (a next version of the lecture notes will describe more detail in a nicer layout).

4.1.1 The OWL family of languages

Purely for legacy purposes, I include here the first three 'species' of OWL (version 1):

- **OWL Lite**, with a classification hierarchy and (relative to OWL DL) simple constraints. While OWL Lite has strong syntactic restrictions, it has only limited semantics restrictions compared to OWL DL[4]. OWL Lite corresponds to the DL $\mathcal{SHIF}(D)$. Putting the DL symbols to the names of the features, we have:
  - Named classes (A)
  - Named properties (P)
  - Individuals ($C(o)$)
  - Property values ($P(o, a)$)
  - Intersection ($C \sqcap D$)
  - Union ($C \sqcup D$)
  - Negation ($\neg C$)
  - Existential value restrictions ($\exists P.C$)
  - Universal value restrictions ($\forall P.C$)
  - Unqualified (0/1) number restrictions ($\geq nP$, $\leq nP$, $= nP$), $0 \leq n \leq 1$
- **OWL DL** had, at the time, 'maximal' expressiveness while maintaining tractability, and has, as the name suggestion, an underlying DL. It has all the features of OWL-lite, and, in addition: Negation, Disjunction, (unqualified) Full cardinality, Enumerated classes, and hasValue. OWL DL corresponds to the DL $\mathcal{SHOIN}(D)$. It has the following features:
  - all of OWL Lite features
  - Arbitrary number restrictions ($\geq nP$, $\leq nP$, $= nP$), with $0 \leq n$
  - Property value ($\exists P.\{o\}$)
  - Enumeration ($\{o_1, ..., o_n\}$)

- **OWL Full**, has a very high expressiveness (losing tractability) and all syntactic freedom of RDF (self-modifying). OWL full has Meta-classes and on can modify

[4]More specifically regarding the latter: negation can be encoded using disjointness and with negation an conjunction, you can encode disjunction. Take, for instance:

```
  Class(C complete unionOf(B C))
```

This is equivalent to

```
  DisjointClasses(notB B)
  DisjointClasses(notC C)
  Class(notBandnotC complete notB notC)
  DisjointClasses(notBandnotC BorC)
  Class(C complete notBandnotC)
```

Table 4.1: Some examples of OWL's construct, the same in DL notation, and an example.

OWL Construct	DL notation	Example
`intersectionOf`	$C_1 \sqcap ... \sqcap C_n$	Human $\sqcap$ Male
`unionOf`	$C_1 \sqcup ... \sqcup C_n$	Doctor $\sqcup$ Lawyer
`complementOf`	$\neg C$	$\neg$Male
`oneOf`	$\{o_1, ..., o_n\}$	{giselle, juan}
`allValuesFrom`	$\forall P.C$	$\forall$hasChild.Doctor
`someValuesFrom`	$\exists P.C$	$\exists$hasChild.Lawyer
`value`	$\exists P.\{o\}$	$\exists$citizenOf.{RSA}
`minCardinality`	$\geq nP$	$\geq$ 2 hasChild
`maxCardinality`	$\leq nP$	$\leq$ 6 enrolledIn

Table 4.2: Some examples of OWL's axioms, the same in DL notation, and an example.

OWL Axiom	DL	Example
`SubClassOf`	$C_1 \sqsubseteq C_2$	Human $\sqsubseteq$ Animal $\sqcap$ Biped
`EquivalentClasses`	$C_1 \equiv ... \equiv C_n$	Man $\equiv$ Human $\sqcap$ Male
`SubPropertyOf`	$P_1 \sqsubseteq P_2$	hasDaughter $\sqsubseteq$ hasChild
`EquivalentProperties`	$P_1 \equiv ... \equiv P_n$	cost $\equiv$ price
`SameIndividual`	$o_1 = ... = o_n$	President_Zuma = J_Zuma
`DisjointClasses`	$C_i \sqsubseteq \neg C_j$	Male $\sqsubseteq$ $\neg$Female
`DifferentIndividuals`	$o_i \neq o_j$	Thabo $\neq$ Andile
`inverseOf`	$P_1 \equiv P_2^-$	hasChild $\equiv$ hasParent$^-$
`transitiveProperty`	$P^+ \sqsubseteq P$	ancestor$^+$ $\sqsubseteq$ ancestor,
		denoted also as Trans(ancestor)
`symmetricProperty`	$P \equiv P^-$	Sym(connectedTo) (common notation)
`functionalProperty`	$\top \sqsubseteq \leq 1P$	$\top \sqsubseteq \leq$ 1hasPresident
`inverseFunctionalProperty`	$\top \sqsubseteq \leq 1P^-$	$\top \sqsubseteq \leq$ 1hasIDNo$^-$

the language. Note that OWL Full is *not* a Description Logic, and we don't look into its details in this course.

As mentioned earlier, OWL and DLs are tightly related, in particular OWL Lite and OWL DL. They have, just like their base DLs, a model theoretic semantics. Like with $\mathcal{SROIQ}$, one can build up more complex representations for the knowledge that has to be represented. Table 4.1 shows a few examples of OWL syntax and its DL counterpart notation.

Compared to most DLs, OWL also incorporates XML Schema datatypes, such as `int`, `string`, `real`, and others; check out the 'data properties' tab in Protégé or the OWL standard to see which ones are supported. Work is under way to incorporate data types better into DL.

While some examples may be familiar by now, let's look at two object property characteristics, being transitivity and reflexivity. Transitivity of a relation R formally means $\forall x, y, z(R(x, y) \land R(y, z) \rightarrow R(x, z))$. For instance, when we have the following two assertions, we can derive the third, provided "part of" is declared transitive (typically abbreviated as Trans(partOf)).

$\star$ your toe is part of your foot

$\star$ your foot is part of your body

$\Rightarrow$ your toe is part of your body

When a relation R is symmetric, it means that 'if a is related to b through relationship

R, then b has the same relation to a'; formally: $\forall x, y(R(x, y) \rightarrow R(y, x))$. For instance, with connectedTo being symmetric, then:

$\star$ Mapungubwe National Park is connected to Northern Thuli Game Reserve
$\Rightarrow$ Northern Thuli Game Reserve is connected to Mapungubwe National Park

We will discuss some other features during the lectures.

4.1.2 OWL syntaxes

There are multiple syntaxes for OWL to meet a variety of requests. Lets start with a few DL axioms and then look at how it turns out in the DL syntaxes:

$\mathsf{firstYearCourse} \sqsubseteq \forall \mathsf{isTaughtBy.Professor}$
$\mathsf{mathCourse} \sqsubseteq \exists \mathsf{isTaughtBy}.\{949352\}$
$\mathsf{academicStaffMember} \sqsubseteq \exists\ \mathsf{teaches.undergraduateCourse}$
$\mathsf{course} \sqsubseteq\ \geq 1\ \mathsf{isTaughtBy}$
$\mathsf{department} \sqsubseteq\ \geq 10\ \mathsf{hasMember} \sqcap \leq 30\ \mathsf{hasMember}$

This is inconvenient for writing emails, webpages, and some people just do not like the symbols. To make it slightly more natural language-like (and easier machine-processable), there is a *functional style syntax*. The same axioms as above then may look like this:

```
Class(firstYearCourse partial restriction(isTaughtBy allValuesFrom
(Professor)))

Class(mathCourse partial restriction(isTaughtBy hasValue (949352)))

Class(academicStaffMember partial restriction(teaches someValuesFrom
(undergraduateCourse)))

Class(course partial restriction(isTaughtBy minCardinality(1)))

Class(department partial restriction(hasMember minCardinality(10))
restriction(hasMember maxCardinality(30)))
```

Then there is the actual *RDF/XML serialization*, i.e., the default computer-processable flavour that is specifically for machine consumption, *not* human consumption. This is how the beginning of the owl file of the example from the OwlGuide looks like:

```
<!ENTITY vin
"http://www.w3.org/TR/2004/REC-owl-guide-20040210/wine#" >
<!ENTITY food
"http://www.w3.org/TR/2004/REC-owl-guide-20040210/food#" > ...
<rdf:RDF
xmlns:vin="http://www.w3.org/TR/2004/REC-owl-guide-20040210/wine#"
xmlns:food="http://www.w3.org/TR/2004/REC-owl-guide-20040210/food#"
... >

<owl:Class rdf:ID="Wine"> <rdfs:subClassOf
rdf:resource="&food;PotableLiquid"/> <rdfs:label
xml:lang="en">wine</rdfs:label> <rdfs:label
xml:lang="fr">vin</rdfs:label> ... </owl:Class>

<owl:Class rdf:ID="Pasta"> <rdfs:subClassOf
rdf:resource="#EdibleThing" /> ... </owl:Class> </rdf:RDF>
```

You are not expected to be able to manually write an ontology in this syntax—there are other syntaxes and tools that generate this serialization—but, depending on the mini-project and any subsequent work in this area, you may have to be able write software that can handle such files.

More examples are included in the lecture slides.

4.2 OWL 2

Over the past 10 years, OWL has been used across subject domains, but in the early years in particular in the health care and life sciences disciplines. Experimentation with the standard revealed expected as well as unexpected shortcomings in addition to the ideas mentioned in the "Future extensions" section of [Horrocks et al., 2003], so that a successor to OWL was deemed to be of value. Work towards a standardization of an OWL 2 took shape after the OWL Experiences and Directions workshop in 2007 and a final draft was ready by late 2008. On October 27 2009 it has become the official OWL 2 W3C recommendation[5]. So, what does OWL 2 consists of, and what does it fix with respect to the OWL standard of 2004?

Limitations of OWL

OWL 2 aims to address the issues described in section 2 of [Cuenca Grau et al., 2008] to a greater or lesser extent, which is neither a superset nor subset of [Horrocks et al., 2003]'s ideas for possible extensions. For instance, the consideration to cater for the Unique Name Assumption did not make it into OWL 2, despite that it has quite an effect on the complexity of a language [Artale et al., 2009]. We briefly summarise the issues; refer to [Cuenca Grau et al., 2008] for details.

Expressivity limitations. In OWL, it is not possible to express qualified cardinality restrictions (e.g., no $\mathsf{Bicycle} \sqsubseteq \geq 2\ \mathsf{hasComponent.Wheel}$), some relational properties were perceived to be missing (notably reflexivity, irreflexivity), limitations on data types (e.g., to be able to express restrictions to a subset of datatype values (ranges) and relationships between values of data properties on one object), and some 'housekeeping' features were missing, such as annotations, imports, versioning, and species validation (see p315 of the paper).

Syntax problems. OWL has both a frame-based legacy (Abstract syntax) and axioms (DL), which was deemed too confusing. For instance, take the following axiom:

```
Class(A partial restriction(hasB someValuesFrom(C))
```

What type of ontology elements do we have? Is `hasB` is data property and `C` a datatype, or is `hasB` an object property and `C` a class? OWL-DL has a strict separation of the vocabulary, but the specification does not precisely specify how to enforce this separation at the syntactic level. In addition, RDF's triple notation is difficult to read and process.

Problems with the semantics. We shall not cover this issue. (For the curious: this has to do with RDF's blank nodes, but unnamed individuals not directly available in $\mathcal{SHOIN}(D)$, and frames and axioms).

Overview of OWL 2

First, take a look at the 'orchestration' of the various aspects of OWL 2 in Figure 4.2. The top section indicates several syntaxes that can be used to serialize the ontology, where RDF/XML is required and the other four are optional. There are mappings between an OWL ontology and RDF graph in the middle, and the lower half depicts that there is both a direct semantics ("OWL 2 DL") and an RDF-based one ("OWL 2 full").

[5]`http://www.w3.org/TR/2009/REC-owl2-overview-20091027/`

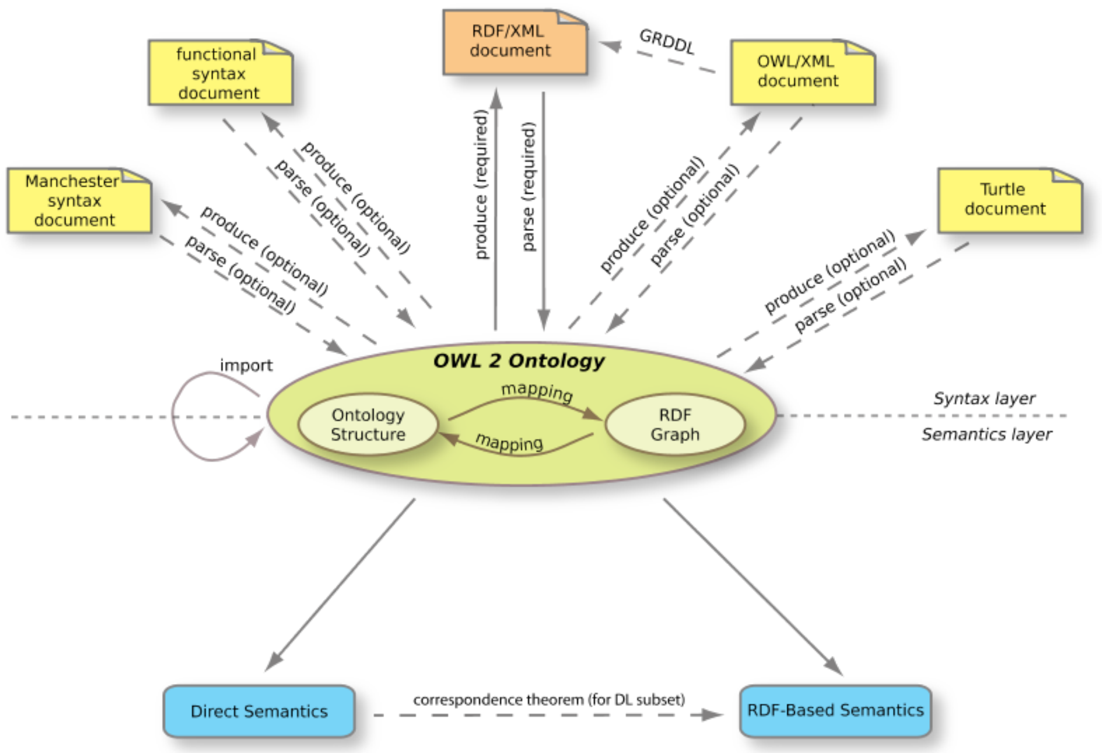


Figure 4.2: Orchestration of syntax and semantics of OWL 2. (Source: `http://www.w3.org/TR/2009/REC-owl2-overview-20091027/`).

Second, the OWL 2 DL species is based on the DL language called $\mathcal{SROIQ}(D)$ [Horrocks et al., 2006], which is more expressive than the base language of OWL-DL ($\mathcal{SHOIN}(D)$) and therewith meeting some of the ontology modellers' requests, such as more properties of properties and qualified number restrictions. There is cleaner support for annotations, debatable (from an ontological perspective, that is) punning for metamodelling, and a 'key' that is not a key in the common and traditional sense of keys in conceptual models and databases. Also, it irons out some difficulties that tool implementers had with the syntaxes of OWL and makes importing ontologies more transparent. To name but a few things, which will be discussed in more detail during the lecture.

Third, in addition to the OWL 2 DL version, there are three OWL 2 profiles[6] [Motik et al., 2009a], which are, strictly speaking, sub-languages of (syntactic restrictions on) OWL 2 DL so as to cater for different purposes of ontology usage in applications. At the time of standardization, they already enjoyed a considerable user base. This choice has its consequences that very well can, but may not necessarily, turn out to be a positive one; this will be explored further in the block on ontology engineering.

The three profiles are:

- **OWL 2 EL**, which is based on the $\mathcal{EL}^{++}$ language [Baader et al., 2005], intended for use with large relatively simple type-level ontologies
- **OWL 2 QL**, which is is based on the DL-Lite$_R$ language [Calvanese et al., 2007], intended for handling and querying large amounts of instances through the ontology, and
- **OWL 2 RL**, which is inspired by Description Logic Programs and pD

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

[Grosof et al., 2003], intended for ontologies with rules and data in the form of RDF triples[7].

Like with OWL 2 DL, each of these languages has automated reasoners tailored to the language so as to achieve the best performance for the application scenario. Indirectly, the notion of the profiles and automated reasoners says you cannot have it all—i.e., very many modelling features—together in one language and expect to have good performance with the ontology and ontology-driven information system. Such is life with the limitations of computers (why this is so, is taught in a theory of computation course), but one can achieve quite impressive results with the languages and its tools that are practically not really doable with paper-based manual efforts or OWL 2 DL. If you are a novice in computational complexity, you may want to consult an informal explanation of trade-offs between ontology languages and language features [Keet and Rodríguez, 2007], and the brief recap in Section 4.3 on the general notion of complexity.

During the lecture, we discuss the OWL 2 features and profiles more comprehensively and provide context to the rationale how and why things ended up the way they did.

4.2.1 OWL 2 features

OWL 2 DL is based on $\mathcal{SROIQ}(D)$ [Horrocks et al., 2006], which we have seen in the DL primer (Section 3.1), which is 2NExpTime-complete [Kazakov, 2008], hence more expressive than OWL-DL ($\mathcal{SHOIN}$, which is NExpTime-complete [Tobies, 2001]). In short, and compared to OWL DL, it has fancier metamodelling and annotations, improved ontology publishing, imports and versioning control. It has a variety of syntaxes, RDF serialization (but no RDF-style semantics). In addition to all the OWL-DL features (recall Section 4.1.1), one can use the following ones in OWL 2 DL as well:

- Qualified cardinality restrictions, $\geq nR.C$ and $\leq nR.C$, semantics:
  - $(\geq nR.C)^{\mathcal{I}} = \{x \mid \sharp\{y \mid (x,y) \in R^{\mathcal{I}} \cap y \in C^{\mathcal{I}}\} \geq n\}$
    In OWL `ObjectMinCardinality(n OPE CE)`; an example in DL notation ≥ 3 hasPart.Door
  - $(\leq nR.C)^{\mathcal{I}} = \{x \mid \sharp\{y \mid (x,y) \in R^{\mathcal{I}} \cap y \in C^{\mathcal{I}}\} \leq n\}$
    In OWL `ObjectMaxCardinality(n OPE CE)`, with DL notation and example ≤ 2 enrolledIn.UGDegree
- Properties of roles:
  - Reflexive (globally): $Ref(R)$, with semantics:
    $\forall x : x \in \Delta^{\mathcal{I}}$ implies $(x,x) \in (R)^{\mathcal{I}}$
  - Reflexive (locally): $\exists R.Self$, with semantics:
    $\{x \mid (x,x) \in R\}$
    In OWL `ObjectHasSelf (OPE)`; e.g., $\exists$knows.Self to state you know yourself.
  - Irreflexive: $Irr(R)$, with semantics:
    $\forall x : x \in \Delta^{\mathcal{I}}$ implies $(x,x) \notin (R)^{\mathcal{I}}$
    For instance, proper parthood is irreflexive: if you heart is a proper part of your body, your body cannot be a proper part of your heart.
  - Asymmetric: Asym(R), with semantics:
    $\forall x,y : (x,y) \in (R)^{\mathcal{I}}$ implies $(y,x) \notin (R)^{\mathcal{I}}$
    For instance, Asym(parentOf): if John is the parent of Divesh, then Divesh cannot be the parent of John.

[7]There are slides with more details and examples of OWL 2 RL, which are from the ESWC'09 tutorial given by Hitzler, Krotzsch, and Rudolf; available at `http://www.semantic-web-book.org/w/images/4/42/OWL2-Rules-Part-2-ESWC09.pdf`

- *Limited* role chaining: e.g., $R \circ S \sqsubseteq R$, with semantics: $\forall y_1, \ldots, y_4 : (y_1, y_2) \in (R)^{\mathcal{I}}$ and $(y_3, y_4) \in (S)^{\mathcal{I}}$ imply $(y_1, y_4) \in (R)^{\mathcal{I}}$, and regularity restriction (strict linear order ("<") on the properties—more details will be discussed in the lecture). For instance: $\mathsf{childOf} \circ \mathsf{childOf} \sqsubseteq \mathsf{grandchildOf}$ so that one can deduce that the child of a child is that person's grandchild.

The tricky part especially in practical ontology development is that some object property features and axioms work only on *simple* object properties, 'simple' meaning that it has no direct or indirect subproperties that are either transitive or are defined by means of property chains; see section 11.1 of the OWL Structural Specification and Functional-Style Syntax[8] [Motik et al., 2009b] for the exact specification of this limitation. Practically, this means that the following features can be used only on simple object properties: ObjectMinCardinality, ObjectMaxCardinality, ObjectExactCardinality, ObjectHasSelf, FunctionalObjectProperty, InverseFunctionalObjectProperty, IrreflexiveObjectProperty, AsymmetricObjectProperty, and DisjointObjectProperties. (The ontology development environment will warn you about it if you use them on a non-simple object property nevertheless, and will prevent you from running the reasoner.)

4.2.2 OWL 2 Profiles

The main rationale for the profiles are computational considerations (see Section 4.3) and robustness of implementations with respect to *scalable* applications. Note: you are not expected to learn the following lists of features by heart (it can be used as a quick 'cheat sheet'), but you do need to know, at least, their intended purpose. The more you practice developing ontologies, the easier it becomes to remember them.

OWL 2 EL

OWL 2 EL is intended for large 'simple' ontologies and focuses on type-level knowledge (TBox). It has a better computational behaviour than OWL 2 DL (polynomial vs. exponential/open). It is based on the DL language $\mathcal{EL}^{++}$ (PTime complete), and it is used for the large medical terminology SNOMED CT. Supported class restrictions:

- existential quantification to a class expression or a data range
- existential quantification to an individual or a literal
- self-restriction
- enumerations involving a single individual or a single literal
- intersection of classes and data ranges

Supported axioms, restricted to allowed set of class expressions:

- class inclusion, equivalence, disjointness
- object property inclusion (w. or w.o. property chains), and data property inclusion
- property equivalence
- transitive object properties
- reflexive object properties
- domain and range restrictions
- assertions
- functional data properties
- keys

NOT supported in OWL 2 EL (with respect to OWL 2 DL):

- universal quantification to a class expression or a data range
- cardinality restrictions

[8] `http://www.w3.org/TR/owl2-syntax/`

- disjunction
- class negation
- enumerations involving more than one individual
- disjoint properties
- irreflexive, symmetric, and asymmetric object properties
- inverse object properties, functional and inverse-functional object properties

OWL 2 QL

OWL 2 QL aims at scenarios for query answering over a large amount of instances with the same kind of performance as relational databases (Ontology-Based Data Access; see Chapter 8). Its expressive features cover several used features of UML Class diagrams and ER models. It is based on $DL\text{-}Lite_{\mathcal{R}}$ (more is possible with the Unique Name Assumption and in some implementations).

The supported axioms in OWL 2 QL take into account what one can use on the left-hand side of the inclusion operator ($\sqsubseteq$, SubClassOf) and what on the right-hand side:

- Subclass expressions restrictions:
  - a class
  - existential quantification (ObjectSomeValuesFrom) where the class is limited to owl:Thing
  - existential quantification to a data range (DataSomeValuesFrom)
- Super expressions restrictions:
  - a class
  - intersection (ObjectIntersectionOf)
  - negation (ObjectComplementOf)
  - existential quantification to a class (ObjectSomeValuesFrom)
  - existential quantification to a data range (DataSomeValuesFrom)

Supported Axioms in OWL 2QL:

- Restrictions on class expressions, object and data properties occurring in functionality assertions cannot be specialized
- subclass axioms
- class expression equivalence (involving subClassExpression), disjointness
- inverse object properties
- property inclusion (not involving property chains and SubDataPropertyOf)
- property equivalence
- property domain and range
- disjoint properties
- symmetric, reflexive, irreflexive, asymmetric properties
- assertions other than individual equality assertions and negative property assertions (DifferentIndividuals, ClassAssertion, ObjectPropertyAssertion, and DataPropertyAssertion)

NOT supported in OWL 2 QL (with respect to OWL 2 DL):

- existential quantification to a class expression or a data range in the subclass position
- self-restriction
- existential quantification to an individual or a literal
- enumeration of individuals and literals
- universal quantification to a class expression or a data range
- cardinality restrictions

– disjunction
– property inclusions involving property chains
– functional and inverse-functional properties
– transitive properties
– keys
– individual equality assertions and negative property assertions

OWL 2 RL

OWL 2 RL's development was motivated by what fraction of OWL 2 DL can be expressed by rules (with equality) and scalable reasoning in the context of RDF(S) application. It uses rule-based technologies (forward chaining rule system, over *instances*) and is inspired by Description Logic Programs and pD*. Reasoning in PTime.

Supported in OWL 2 RL:

- More restrictions on class expressions (see table 2 of [Motik et al., 2009a], e.g. no SomeValuesFrom on the right-hand side of a subclass axiom)
- All axioms in OWL 2 RL are constrained in a way that is compliant with the restrictions in Table 2.
- Thus, OWL 2 RL supports all axioms of OWL 2 apart from disjoint unions of classes and reflexive object property axioms.
- No $\forall$ and $\neg$ on the left-hand side, and $\exists$ and $\sqcup$ on right-hand side of $\sqsubseteq$

4.2.3 OWL 2 syntaxes

As mentioned in Section 4.1.2, there are more syntaxes for OWL 2 than for OWL. Let us take as example the DL axiom

$\mathsf{FirstYearCourse} \sqsubseteq \forall \mathsf{isTaughtBy.Professor}$

Rendering this in RDF/XML yields:

```
<!-- http://www.semanticweb.org/ontologies/2011/10/exOKB12.owl#FirstYearCourse -->

<owl:Class rdf:about="&exOKB12;FirstYearCourse">
    <rdfs:subClassOf rdf:resource="&owl;Thing"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="&exOKB12;isTaughtBy"/>
            <owl:allValuesFrom rdf:resource="&exOKB12;Professor"/>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
```

This RDF/XML fragment tells us that the ontology is called `exOKB12` (abbreviated name for the full URI), `FirstYearCourse` is a `subClassOf` the root-class `Thing`, and of the restriction on `FirstYearCourse`, being that the restriction is `owl:onProperty` object property `isTaughtBy` and the 'filler', i.e., to which the restriction applies, is `allValuesFrom` (i.e., $\forall$) `Professor`.

In OWL/XML (also not intended for human consumption), we have the same as follows:

```
<SubClassOf>
    <Class IRI="#FirstYearCourse"/>
    <Class abbreviatedIRI="owl:Thing"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="#FirstYearCourse"/>
    <ObjectAllValuesFrom>
```

```
            <ObjectProperty IRI="#isTaughtBy"/>
            <Class IRI="#Professor"/>
        </ObjectAllValuesFrom>
    </SubClassOf>
```

The functional syntax equivalent is as follows:

```
Declaration(Class(:FirstYearCourse))
SubClassOf(:FirstYearCourse owl:Thing)
SubClassOf(:FirstYearCourse ObjectAllValuesFrom(:isTaughtBy :Professor))
```

The Manchester syntax rendering is intended exceedingly for human reading, for non-logicians, and for ease of communication in, say, emails that do not render mathematical symbols well. On the one hand, there is a Protégé-generated Manchester syntax rendering:

```
Class: <http://www.semanticweb.org/ontologies/2011/10/exOKB12.owl#FirstYearCourse>

    SubClassOf:
        owl:Thing,
        <http://www.semanticweb.org/ontologies/2011/10/exOKB12.owl#isTaughtBy> only
            <http://www.semanticweb.org/ontologies/2011/10/exOKB12.owl#Professor>
```

But this usually gets abbreviated as follows:

```
Class: FirstYearCourse
SubClassOf:
        owl:Thing,
        isTaughtBy only Professor
```

or, even shorter:

```
FirstYearCourse SubClassOf isTaughtBy only Professor
```

There are several really non-standard representations of OWL ontologies for various reasons, such as interface design and making it easier for non-logicians to contribute to ontology development. For instance, in pseudo-natural language with ACE (which we will look at later in Chapter 7), and graphical renderings, like with Ontograf and depicted in Figure 4.3, where the axiom shows when hovering over the coloured line representing the object property.

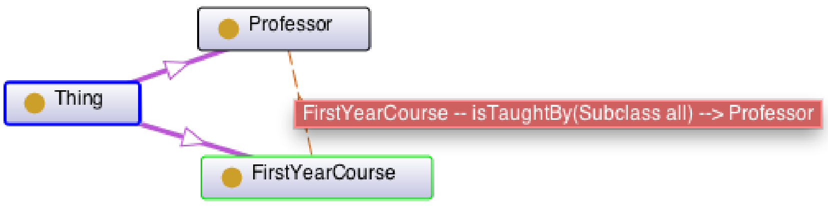


Figure 4.3: Screenshot of the " FirstYearCourse $\sqsubseteq \forall$isTaughtBy.Professor" in the Ontograf plugin for the Protégé ontology development environment; the axiom appears when hovering over the coloured dashed line representing the object property.

4.3 Computational Properties

We have seen different 'species' of OWL, which have more or less language features, and that this was motivated principally by scalability issues of the very expressive languages. Different languages/problems have different complexity (NP-complete, PSPACE, EXPTIME etc.). Section 4.3.1 contains a very brief recap on complexity, and the specifics for OWL are summarised in Section 4.3.2.

4.3.1 Complexity recap

Theory of computation concerns itself with, among other things, languages and its dual, problems. A problem is the question of deciding whether a given string is a member of some particular language; more precisely: if Σ is an alphabet, L is a language over Σ, then the problem L is "given a string $w \in \Sigma^*$, decide whether or not w is in L". The usage of 'problem' and 'language' is interchangeable. When we focus on strings for their own sake (e.g., in the set $\{o^n 1^n \mid n \geq 1\}$), then we tend to think of the set of strings as a language. When we focus on the 'thing' that is encoded as a string (e.g., a particular graph, a logical expression, satisfiability of a class), we tend to think of the set of strings as a problem. Within the context of ontologies, we typically talk of the representation languages and reasoning problems.

There are several classes of languages; see Figure 4.4. The regular free languages have their counterpart with finite automata; the context-free languages with push-down automata; the recursive languages is the class of languages accepted by a Turing machine (TM) that always halts; the recursively enumerable languages is the class of languages defined by a TM; the non-recursively enumerable languages is the class of languages for which there is no TM (e.g., the diagonalization language). The recursive languages, and, to a lesser extent, the recursively enumerable languages, are by far the most interesting ones for ontologies.

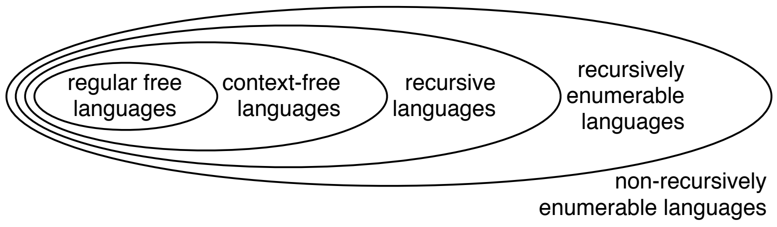

regular free languages
context-free languages
recursive languages
recursively enumerable languages
non-recursively enumerable languages

Figure 4.4: Graphical depiction of the main categories of languages.

Turing machines are used as a convenient abstraction of actual computers for the notion of computation. *A TM that always halts = algorithm*, i.e., the TM halts on all inputs in finite time, either accepting or rejecting; hence, the recursive languages are *decidable* problems/languages. Problems/languages that are not recursive are called *undecidable*, and they do not have an algorithm; if they are in the class of recursively enumerable languages (but not recursive), then they have a *procedure that runs on an arbitrary TM* that may give you an answer but may very well never halt; see also Figure 4.5. First order predicate logic in its full glory is undecidable. Description logics are decidable fragments of first order predicate logic, i.e., they are recursive languages and (can) have algorithms.

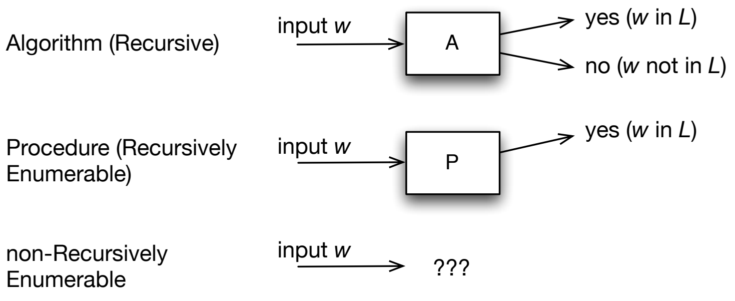


Figure 4.5: Graphical depiction of the main categories of languages; the rectangle denotes a Turing Machine; w is a string and L is a language.

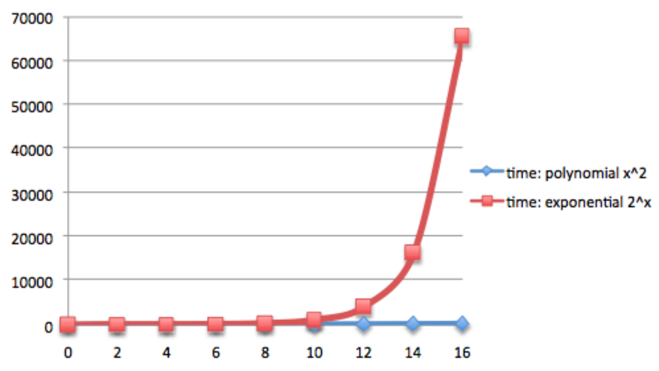


Figure 4.6: General idea of time complexity of an algorithm, as function of the size of the input. e.g.: All basic arithmetic operations can be computed in polynomial time; evaluating a position in generalized chess and checkers on an $n \times n$ board costs exponential time.

Not all algorithms are alike, however, and some take up more time (by the CPU) or space (in the form of memory size) to compute the answer than others. So, we want to know for a given problem, the answer to *"how much time [/space] does it take to compute the answer, as a function of the size of the input?"*. If the computation takes many years with the top-of-the-range hardware, then it is still not particularly interesting to implement (from a computer science viewpoint, that is). To structure these matters, we use the notion of a *complexity class*. There are very many of them, but we only refer to a few in the context of ontologies. For instance, it may take a polynomial amount of time to compute class subsumption for an OWL 2 EL-formalised ontology and exponential time to compute satisfiability of an EER diagram (represented in the DL $\mathcal{DLR}_{\mathsf{ifd}}$) and the bigger the diagram (more precisely: the logical theory), the longer it takes. The intuition is depicted in Figure 4.6: for small ontologies, there is but a minor difference in performance, but one really starts to notice it with larger logical theories. Looking ahead at the complexity classes relevant for OWL, we list here a description of the meaning of them (copied from the OWL 2 Profiles Standard page [Motik et al., 2009a]):

- **Decidability open** means that it is not known whether this reasoning problem is decidable at all.
- **Decidable, but complexity open** means that decidability of this reasoning problem is known, but not its exact computational complexity. If available, known lower bounds are given in parenthesis; for example, (NP-Hard) means that this problem is at least as hard as any other problem in NP.
- **X-complete** for X one of the complexity classes explained below indicates that tight complexity bounds are known—that is, the problem is known to be both in the complexity class X (i.e., an algorithm is known that only uses time/space in X) and hard for X (i.e., it is at least as hard as any other problem in X). The following is a brief sketch of the classes used in this table, from the most complex one down to the simplest ones.
    - **2NEXPTIME** is the class of problems solvable by a nondeterministic algorithm in time that is at most double exponential in the size of the input (i.e., roughly 2^{2^n}, for n the size of the input).
    - **NEXPTIME** is the class of problems solvable by a nondeterministic algorithm in time that is at most exponential in the size of the input (i.e., roughly 2^n, for n the size of the input).
    - **PSPACE** is the class of problems solvable by a deterministic algorithm using space that is at most polynomial in the size of the input (i.e., roughly n^c, for

n the size of the input and c a constant).

- **NP** is the class of problems solvable by a nondeterministic algorithm using time that is at most polynomial in the size of the input (i.e., roughly n^c, for n the size of the input and c a constant).
- **PTIME** is the class of problems solvable by a deterministic algorithm using time that is at most polynomial in the size of the input (i.e., roughly n^c, for n the size of the input and c a constant). PTIME is often referred to as tractable, whereas the problems in the classes above are often referred to as intractable.
- **LOGSPACE** is the class of problems solvable by a deterministic algorithm using space that is at most logarithmic in the size of the input (i.e., roughly $log(n)$, for n the size of the input and c a constant). NLOGSPACE is the nondeterministic version of this class.
- **AC^0** is a proper subclass of LOGSPACE and defined not via Turing Machines, but via circuits: AC^0 is the class of problems definable using a family of circuits of constant depth and polynomial size, which can be generated by a deterministic Turing machine in logarithmic time (in the size of the input). Intuitively, AC^0 allows us to use polynomially many processors but the run-time must be constant. A typical example of an AC^0 problem is the evaluation of first-order queries over databases (or model checking of first-order sentences over finite models), where only the database (first-order model) is regarded as the input and the query (first-order sentence) is assumed to be fixed. The undirected graph reachability problem is known to be in LogSpace, but not in AC^0.

4.3.2 OWL and computational complexity

In this setting, we are interested in the following reasoning problems: ontology consistency, class expression satisfiability, class expression subsumption, instance checking, and (Boolean) conjunctive query answering (recall Section 3.3). When evaluating complexity, the following parameters are considered (copied from section 5 of the OWL 2 Profiles standard [Motik et al., 2009a]):

- **Data Complexity**: the complexity measured with respect to the total size of the assertions in the ontology.
- **Taxonomic Complexity**: the complexity measured with respect to the total size of the axioms in the ontology.
- **Query Complexity**: the complexity measured with respect to the total size of the query.
- **Combined Complexity**: the complexity measured with respect to both the size of the axioms, the size of the assertions, and, in the case of conjunctive query answering, the size of the query as well.

Table 4.3 summarizes the known complexity results for OWL 2 under both RDF and the direct semantics, OWL 2 EL, OWL 2 QL, OWL 2 RL, and OWL 1 DL. The results refer to the *worst-case complexity* of these reasoning problems and, as such, do not say that implemented algorithms necessarily run in this class on all input problems, or what space/time they use on some/typical/certain kind of problems. For X-complete problems, these results only say that a reasoning algorithm cannot use less time/space than indicated by this class on all input problems, where "X" is one of the complexity classes listed in the previous section.

Table 4.3: Complexity of OWL species (Source: [Motik et al., 2009a]).

Language	**Reasoning Problems**	**Taxonomic Complexity**	**Data Complexity**	**Query Complexity**	**Combined Complexity**
OWL 2 RDF-Based Semantics	Ontology Consistency, Class Expression Satisfiability, Class Expression Subsumption, Instance Checking, Conjunctive Query Answering	Undecidable	Undecidable	Undecidable	Undecidable
OWL 2 Direct Semantics	Ontology Consistency, Class Expression Satisfiability, Class Expression Subsumption, Instance Checking	2NEXPTIME-complete (NEXPTIME if property hierarchies are bounded)	Decidable, but complexity open (NP-Hard)	Not Applicable	2NEXPTIME-complete (NEXPTIME if property hierarchies are bounded)
	Conjunctive Query Answering	Decidability open	Decidability open	Decidability open	Decidability open
OWL 2 EL	Ontology Consistency, Class Expression Satisfiability, Class Expression Subsumption, Instance Checking	PTIME-complete	PTIME-complete	Not Applicable	PTIME-complete
	Conjunctive Query Answering	PTIME-complete	PTIME-complete	NP-Complete	PSPACE-Complete
OWL 2 QL	Ontology Consistency, Class Expression Satisfiability, Class Expression Subsumption, Instance Checking,	NLogSpace-complete	in AC^0	Not Applicable	NLogSpace-complete
	Conjunctive Query Answering	NLogSpace-complete	in AC^0	NP-complete	NP-complete
OWL 2 RL	Ontology Consistency, Class Expression Satisfiability, Class Expression Subsumption, Instance Checking,	PTIME-complete	PTIME-complete	Not Applicable	PTIME-complete
	Conjunctive Query Answering	PTIME-complete	PTIME-complete	NP-complete	NP-complete
OWL DL	Ontology Consistency, Class Expression Satisfiability, Class Expression Subsumption, Instance Checking,	NEXPTIME-complete	Decidable, but complexity open (NP-Hard)	Not Applicable	NEXPTIME-complete
	Conjunctive Query Answering	Decidability open	Decidability open	Decidability open	Decidability open

The interested reader may want to have a look at the Description Logic Complexity Navigator[9] to see how other combinations of language features fare in the complexity (of concept satisfiability and ABox consistency), and more details can be found in [Baader et al., 2008].

4.4 Exercises

Exercise 7. Install Protégé 4.x from the Protégé website at `http://protege.stanford.edu/overview/protege-owl.html`, if not already installed on your computer, and acquaint yourself with the software by going through the Pizza Ontology Tutorial (can be downloaded online, or from the course's Vula page). Note that the Pizza tutorial is mainly intended to acquaint yourself with the tool, not that it is a cookbook for best practices in ontology development.

Exercise 8. Create a new ontology, add the vegan and vegetarian from last week's exercise, and check both $\mathcal{O} \vdash Vegan \sqsubseteq Vegetarian$ and $\mathcal{O} \vdash Vegetarian \sqsubseteq Vegan$. Describe the outcomes.

Exercise 9. Have a look at another ontology development environment: MoKi (at `https://moki.fbk.eu`), which uses a semantic wiki.

1. Repeat the previous exercise.
2. Compare the tools by considering, among others: do they both support OWL 2 DL? Which one is easier to navigate? Which one has the most features to help ontology development? Which one is easier for a collaborative ontology development project?

Exercise 10. Describe in your own words the motivations to develop OWL 2.

Exercise 11. What are the new features in OWL 2 DL compared to OWL-DL?

Exercise 12. What are the new features in OWL 2 DL compared to OWL-DL?

Exercise 13. Complete Table 4.4: Verify the question marks in the table (tentatively all "–"), fill in the dots, and any "±" should be qualified at to what the restriction is. You may prefer to distribute this exercise among your class mates.

Exercise 14. Consider some medical ontology. You know that an injury (a cut, a fracture) to a bone in your hand is also an injury to your hand. How can you model this, and similar, information in an OWL 2 DL ontology such that it infers this not only for injuries to hands, but for any injury to any anatomical body part to an injury to its (direct/indirect) whole? Which OWL 2 DL feature do you need for this?

Exercise 15. Load `university1.owl` (note the OWL species) in Protégé and try to represent:

a. A `Joint Honors Maths & Computer Science Student`, who is one who takes both Computer Science and Mathematics modules.
b. A `Single Honours Maths Student` (or [Computer Science, Economics]) is one who takes only Maths [Computer Science, Economics] modules.

Is it possible? If yes, how, if not, why not?

Exercise 16. Classify the ontology of the previous question, and describe what happened and changed.

[9]`http://www.cs.man.ac.uk/~ezolin/dl/`

Table 4.4: Partial comparison of some OWL features

Language ⇒	OWL 1		OWL 2	OWL 2 Profiles		
Feature ⇓	**Lite**	**DL**	**DL**	EL	QL	RL
Role hierarchy	+	+	+	.	+	.
N-ary roles (where $n \geq 2$)	−	−	−	.	?	.
Role chaining	−	−	+	.	−	.
Role acyclicity	−	−	−	.	−	.
Symmetry	+	+	+	.	+	.
Role values	−	−	−	.	−	.
Qualified number restrictions	−	−	+	.	−	.
One-of, enumerated classes	?	+	+	.	−	.
Functional dependency	+	+	+	.	?	.
Covering constraint over concepts	?	+	+	.	−	.
Complement of concepts	?	+	+	.	+	.
Complement of roles	−	−	+	.	+	.
Concept identification	−	−	−	.	−	.
Range typing	−	+	+	.	+	.
Reflexivity	−	−	+	.	−	.
Antisymmetry	−	−	−	.	−	.
Transitivity	+	+	+	.	−	.
Asymmetry	?	?	+	−	+	+
Irreflexivity	−	−	+	.	−	.
.	.	.	.	.	.	.

Exercise 17. The university has a regulation that each undergraduate student must take exactly 2 modules. How might you model this restriction the the ontology of the previous question?

Exercise 18. `Student 9` takes `MT101`, `CS101`, and `CS102`. Do you think your ontology is consistent? Describe why. Check your answer by adding the student and his courses, run the reasoner and examine the inferences (in yellow).

Exercise 19. `Student 10` takes `MT101`, `CS101`, and `EC101`. Do you think your ontology is consistent? Describe why. Check your answer by adding the data, running the reasoner, and examining the inferences.

Exercise 20. Open the `computerscience.owl`, find the principal errors in the ontology, and distinguish them from the 'knock-on' errors that are merely a consequence of the principal errors.

Exercise 21. What would you propose to a modeller how to fix it, and why? Note that "fixing" is to be understood as obtaining a satisfiable ontology other than just deleting the unsatisfiable classes.

4.5 Literature and reference material

1. Ian Horrocks, Peter F. 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, 2003.

2. OWL Guide: `http://www.w3.org/TR/owl-guide/`
3. OWL Reference: `http://www.w3.org/TR/owl-ref/`
4. OWL Abstract Syntax and Semantics: `http://www.w3.org/TR/owl-semantics/`
5. B. Cuenca Grau, I. Horrocks, B. Motik, B. Parsia, P. Patel-Schneider, and U. Sattler. OWL 2: The next step for OWL. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, 6(4):309-322, 2008.
6. Pascal Hitzler, Markus Kroetzsch, Sebastian Rudolph. *Foundations of Semantic Web Technologies*. Chapman & Hall/CRC, 2009, 455p.
7. OWL 2 quick Reference: `http://www.w3.org/TR/owl2-quick-reference/`
8. OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax: `http://www.w3.org/TR/owl2-syntax/`
9. OWL 2 Web Ontology Language Profiles: `http://www.w3.org/TR/owl2-profiles/`

Part II

Ontology development

CHAPTER 5

Methods and Methodologies

In the previous block we looked at languages for representing ontologies, and got experience in reading existing ontologies, adding and removing some axioms, and the automated reasoner. But how exactly did someone come up with the whole ontology in the first place? What can, or should, you do when you have to develop your own ontology? Just like in software engineering, there are methods and methodologies to guide you through it, or at least help out with one of the steps in the development of an ontology.

There is, however, not just one way of doing it or a single up-to-date comprehensive *methodology* for ontology development that covers everything you possibly probably need. There are several leaner proposals along the line of generic 'waterfall' and 'agile' approaches, inspired by software development methodologies, which are at the level of general guidelines and more and less detailed stages, which we shall cover in this chapter. Diagrammatically, such (generalised!) methodologies have the tasks as shown in Figure 5.1: this one may look like a 'waterfall', but practically, it can be an iterative not only within the ontology development, but also that "maintenance" may involve substantial redesign or adding a new module that follows those development steps again.

There are also many *methods* and guidelines, which to a greater or lesser extent can form part of a comprehensive ontology development process. A main reason for this state of affairs is that there is still much to be done, and there are many different use-case scenarios.

We will build up from methods to methodologies in this chapter, and go into some detail of the 'top-down' and 'bottom-up' approaches in the next two chapters. Practically, this means we start with high-level methodologies in Section 5.1, then a sampling of a few methods that can be used as a component within those methodologies (Section 5.2), and then point out there are dependencies along the process, looking at some consequences choices one can make (Section 5.3).

5.1 Methodologies for ontology development

Several specific methodologies for ontology development exist following more or less the general idea depicts din Figure 5.1, notably METHONTOLOGY [Fernández et al., 1999], MOKI [Ghidini et al., 2009], On-To-Knowledge

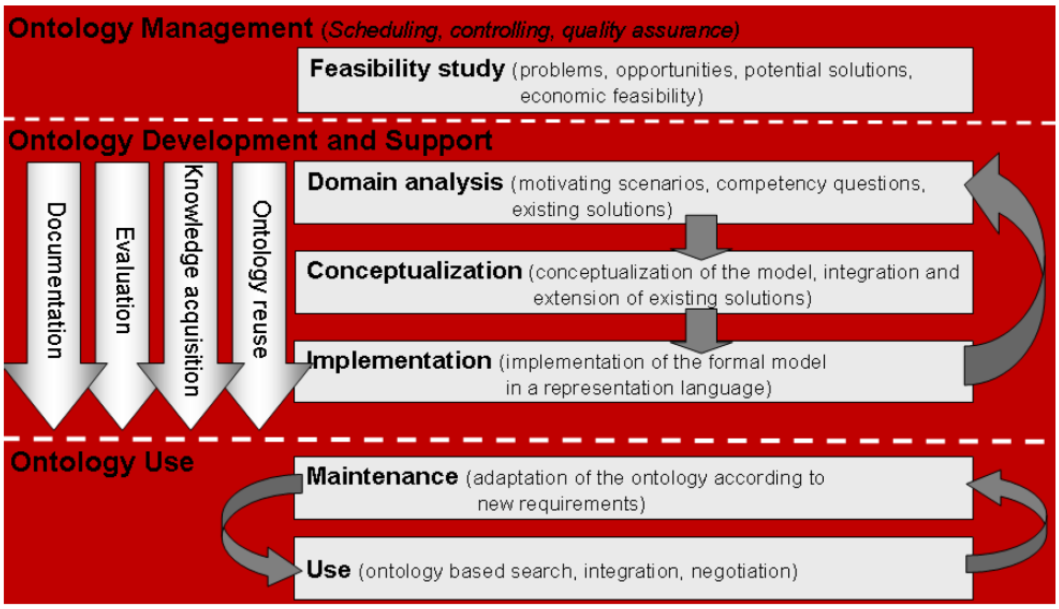

Ontology Management (Scheduling, controlling, quality assurance)
Feasibility study (problems, opportunities, potential solutions, economic feasibility)
Ontology Development and Support
Documentation
Evaluation
Knowledge acquisition
Ontology reuse
Domain analysis (motivating scenarios, competency questions, existing solutions)
Conceptualization (conceptualization of the model, integration and extension of existing solutions)
Implementation (implementation of the formal model in a representation language)
Ontology Use
Maintenance (adaptation of the ontology according to new requirements)
Use (ontology based search, integration, negotiation)

Figure 5.1: Main tasks in ontology engineering (Source: [Simperl et al., 2010])

[Staab et al., 2001], the NeON methodology [Suarez-Figueroa et al., 2008], Melting Point methodology [Garcia et al., 2010], OntoSpec [Kassel, 2005], DiDOn [Keet, 2012b], and the “Ontology Development 101” (OD101) [Noy and McGuinness, 2001]. They are not simply interchangeable in that one could pick any one of them and it will work out well. Besides that some are older (outdate, perhaps, by now) than others, they can be distinguished in core approach, being between:

- micro-level ontology authoring versus a macro-level systems-view of ontology development;
- isolated, single, stand-alone, ontology development versus collaborative development of ontologies and ontology networks.

Micro-level methodologies focus on the viewpoint of the details emphasising formalisation aspects, which goes into *ontology authoring*, for it is about writing down the actual axioms and design choices sometimes even driven by the language. Macro-level methodologies, on the other hand, emphasise the processes from an information systems and IT viewpoint, such as depicted in Figure 5.1. They may merge into comprehensive methodologies in the near future, but are not yet at present.

Regarding the second difference, this reflects a division between ‘old’ and ‘new’ methodologies in the sense that the older ones assume a setting that was typical of 10-20 years ago: the development of a single monolithic ontology by one or a few people in one location. The more recent ones take into account the changing landscape in ontology development over the years, being towards collaboratively building ontology networks that cater for characteristics such as *dynamics, context, collaborative, and distributed development*. For instance, domain experts and knowledge engineers may work on an ontology simultaneously from two different locations, and the automated reasoning may well be distributed over other locations with more powerful machines.

The remainder of this section provides an overview of these two types of guidelines, and closes with a few notes on tools.

5.1.1 Macro-level development methodologies

The macro-level methodologies all will get you started with domain ontology development in a structured fashion, albeit not all in the exact same way, and sometimes that is even intended like that. For instance, one may commence with a feasibility study and assessment of potential economic benefits of the ontology-driven approach to solving the problem(s) at hand, or assume that is sorted out already or not necessary and commence with the actual development methodology by conducting a requirements analysis of the ontology itself and/or find and describe case studies. A well-known instantiation of the generic notions of the development process depicted in Figure 5.1, is the comparatively comprehensive Methontology methodology [Gómez-Pérez et al., 2004], which has been applied to various subject domains since its development in the late 1990s (e.g., the chemicals [Fernández et al., 1999] and legal domain [Corcho and Mariano Fernández-López, 2005]). This methodology is for single ontology development and while several practicalities are superseded with more recent [Corcho et al., 2003] and even newer languages, tools, and methodologies [Suarez-Figueroa et al., 2008], the core procedure still holds. The five main steps are:

1) specification,
2) conceptualization with intermediate representations such as in text or diagrams
3) formalization,
4) implementation, and
5) maintenance

In addition, there are various supporting tasks, such as documentation and version control. Ontology management may vary somewhat across the methodologies, such as helping with development of a Gantt chart for several ontology development scenarios. A refinement over the years is, among others, the better provision of 'intermediate representations'; e.g., the MOdelling wiKI MoKi[1] has a specific feature for automatic translation between formal and semi or informal specifications by the different experts, and is even tailored to provide an intermediate representation interface to let domain experts, ontologists, and logicians work together on a single project.

Methontology is, practically, superseded by the NeON methodology. Instead of the neat, straight-forward five steps, there are many steps; see Figure 5.2. Various development scenarios are then specified by combining a subset of those steps and in some order, which then results in different planning of the ontology activities.

Not even the NeON methodology covers all options—i.e., all the possible permutations at each step—that should be in an ontologist's 'tool box', though. For instance, some mention "non-ontological resource reuse" for bottom-up ontology development (number 2 in Figure 5.2), and note NLP and reuse of thesauri, but lack detail on how this is to be done—for that, one has to search the literature and look up specific methods and tools and the other bottom-up routes mentioned in Chapter 7 that can or have to be 'plugged in' the methodology actually being applied. A glaring absence from the methodologies is that none of them incorporates a 'top-down' step on foundational ontology use [Garcia et al., 2010] to enforce precision and interoperability with other ontologies and reuse generic classes and object properties to facilitate domain ontology development. We will look at this in some detail Chapter 6. For the older methodologies this may be understandable, give that at the time they were hardly available, but it is a missed opportunity for the more recent methodologies.

[1]the MOdelling wiKI MOKI `http://moki.fbk.eu/` was developed during the APOSDLE project[2] for work-integrated learning [Ghidini et al., 2009])

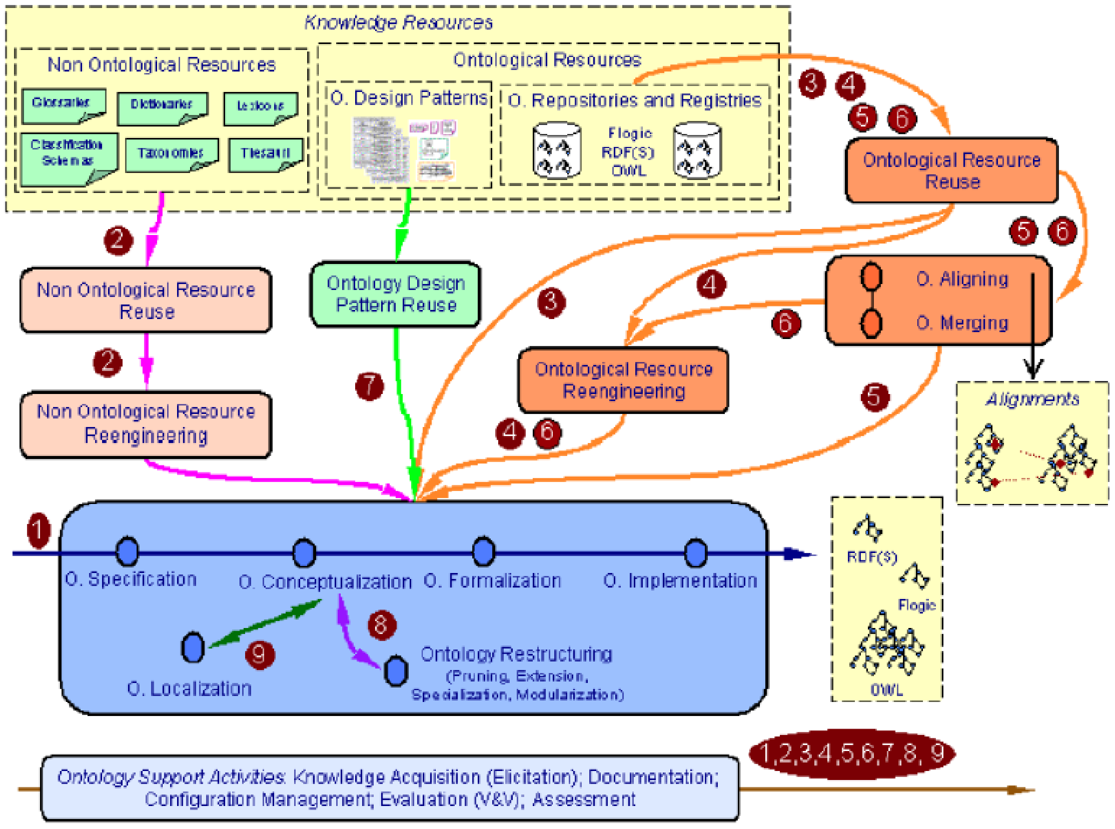

Knowledge Resources
Non Ontological Resources
Glossaries
Dictionaries
Lexicons
Classification Schemas
Taxonomies
Thesauri
Ontological Resources
O. Design Patterns
O. Repositories and Registries
Flogic
RDF(S)
OWL
Ontological Resource Reuse
Non Ontological Resource Reuse
Ontology Design Pattern Reuse
O. Aligning
O. Merging
Alignments
Ontological Resource Reengineering
Non Ontological Resource Reengineering
O. Specification
O. Conceptualization
O. Formalization
O. Implementation
O. Localization
Ontology Restructuring
(Pruning, Extension, Specialization, Modularization)
RDF(S)
Flogic
OWL
1,2,3,4,5,6,7,8, 9
Ontology Support Activities: Knowledge Acquisition (Elicitation); Documentation; Configuration Management; Evaluation (V&V); Assessment

Figure 5.2: Graphical depiction of several different steps in ontology development, where each step has its methods and interactions with other steps (Source: [Suarez-Figueroa et al., 2008])

A useful recent addition to the ontology development methodology landscape is the Ontology Summit 2013 Communiqué's[3] take on the matter with the *ontology lifecycle model*; see Figure 5.3. Each stage has its own set of questions that ought to be answered satisfactorily. To give you a flavour of those questions that need to be answered in an ontology development project, I include here random selection of such questions at each stage, which also address evaluation of the results of that stage (see the communiqué for more of such question):

- Requirements development phase
  - Why is this ontology needed? (What is the rationale? What are the expected benefits)?
  - Are there existing ontologies or standards that need to be reused or adopted?
  - What are the competency questions? (what questions should the ontology itself be able to answer?)
- Ontological analysis phase
  - Are all relevant terms from the use cases documented?
  - Are all entities within the scope of the ontology captured?
- System design phase
  - What operations will be performed, using the ontology, by other system components? What components will perform those operations? How do the business requirements identified in the requirements development phase apply to those specific operations and components?

[3]`http://ontolog.cim3.net/cgi-bin/wiki.pl?OntologySummit2013_Communique`

- How will the ontology be built, evaluated, and maintained? What tools are needed to enable the development, evaluation, configuration management, and maintenance of the ontology?

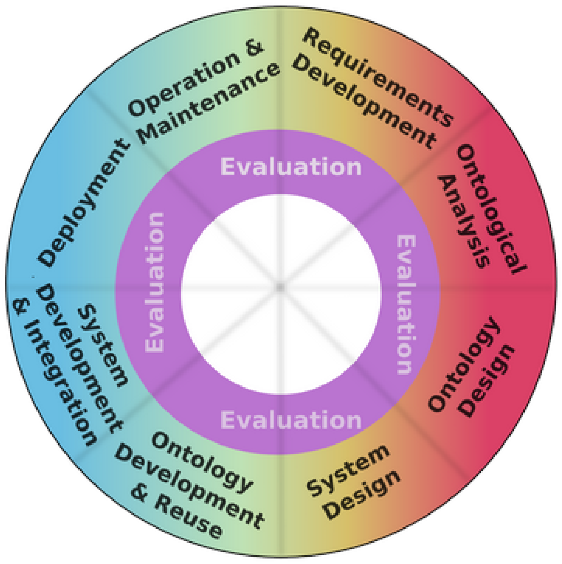


Figure 5.3: Ontology Summit 2013's lifecycle model (Source: `http://ontolog.cim3.net/cgi-bin/wiki.pl?OntologySummit2013_Communique`)

It is beyond the current scope to provide a comparison of the methodologies (see for a recent overview [Garcia et al., 2010]). Either way, it is better to pick one of them to structure your activities for developing a domain ontology than using none at all. Using none at all amounts to re-inventing the wheel and stumbling upon the same difficulties and making the same mistakes developers have made before, but a good engineer has learned from previous mistakes. The methodologies aim to prevent common mistakes and omission, and lets you to carry out the tasks better than otherwise would have occurred without using one.

5.1.2 Micro-level development

OntoSpec, OD101, and DiDOn can be considered 'micro-level' methodologies: they focus on guidelines to formalise the subject domain, i.e., providing guidance *how* to go from an informal representation to a logic-based one. While this could be perceived to be part of the macro-level approach, as it happens, such a 'micro-level view' actually does affect some macro-level choices and steps. It encompasses not only axiom choice, but also other aspects that affects that, such as:

1) Requirements analysis, with an emphasis on purpose, use cases regarding expressiveness (temporal, fuzzy, n-aries etc.), types of queries, reasoning services needed;
2) Design an ontology architecture, such as modular, and if so, in which way, distributed or not, which (logic-based) framework to use;
3) Choose principal representation language and consider encoding peculiarities;
4) Consider and choose a foundational ontology and make modelling decisions (e.g., on attributes and n-aries as relations or classes);
5) Consider domain, top-domain level, ontology design pattern ontology reuse, if applicable, and any ontology matching technique required as a result of that;
6) Consider semi-automated bottom-up approaches, tools, and language transformations, and remodel if needed to match the decisions in steps 3 and 4;

7) Formalization (optionally with intermediate representations), including:
    a) examine and add the classes, object properties, constraints, rules taking into account the imported ontologies;
    b) use an automated reasoner for debugging and detecting anomalous deductions in the logical theory;
    c) use ontological reasoning services for ontological quality checks (e.g., OntoClean and RBox Compatibility);
    d) add annotations;
8) Generate versions in other ontology languages, 'lite' versions, etc, if applicable;
9) Deployment, with maintenance, updates, etc.

Some of them are incorporated also in the macro-level methodologies, but do not yet clearly feature in the detail required for authoring ontologies. There is much to say about these steps, and even more yet to be investigated and developed (and they will be revised and refined in due time), and some more detail and an example can be found in [Keet, 2012b]. Let us look briefly at the language choice and some modelling choices on formalising it, in order to demonstrate that the 'micro' is not as as micro 'single step' as it initially might seem, and that the micro aren't simply small trivial choices to make in the development process.

The representation language

Regarding formalisation, the first aspect is to choose a suitable logic-based language, which ought to be the optimal choice based on the language and automated reasoning requirements (if any), that, in turn, ought to follow from the overall purpose of the ontology (due to computational limitations), if there is a purpose at all [Keet, 2010a]. Generalising a bit, they fall broadly into two main categories: light-weight ontologies—hence, languages—to be deployed in systems for, among others, annotation, natural language processing, and ontology-based data access, and there are 'scientific ontologies' for representing the knowledge of a subject domain (e.g., the Foundational Model of (human) Anatomy [Rosse and Mejino Jr, 2003], BioPax for biological pathways [Demir et al., 2010], data mining and optimisation [Hilario et al., 2011, Keet et al., 2015]). More importantly for choosing the suitable language, is that the first group of ontologies require support for navigation, simple queries to retrieve a class in the hierarchy, and scalability. Thus, a language with low expressiveness suffices, such as the Open Biological and biomedical Ontologies' `obo`-format, the W3C standardised Simple Knowledge Organisation System (SKOS) language (essentially RDF) [Miles and Bechhofer, 2009], and the OWL 2 EL or OWL 2 QL profile [Motik et al., 2009a]. For a scientific ontology, on the other hand, we need a very expressive language to capture fine-grained distinctions between the entities. This also means one needs (and can use fruitfully) more reasoning services, such as satisfiability, classification, and complex queries. One can choose any language, be it full first order predicate logic with or without an extension (e.g., a temporal extension), or one of the very expressive OWL species to guarantee termination of the reasoning services and foster interoperability and reuse with other ontologies. The basic idea is summarised in Figure 5.4, which is yet to be refined further with more ontology languages, such as the OWL 2 RL profile or SWRL for rules and the $\mathcal{DLR}$ family of DL languages that can handle n-ary relationships (with $n \geq 2$) properly.

The analysis of the language aspects can be pushed further, and one may wish to consider the language in a more fine-grained way and prefer one semantics over another and one ontological commitment over another. For instance, assessing whether one needs access to the components of a relationship, the need for n-aries, or whether asymmetry is

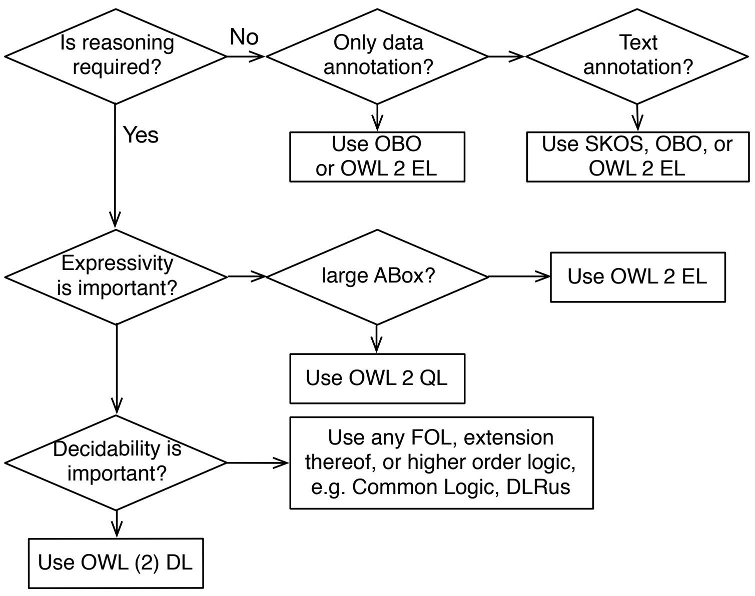

Is reasoning required?
No
Only data annotation?
Text annotation?
Use OBO or OWL 2 EL
Use SKOS, OBO, or OWL 2 EL
Yes
Expressivity is important?
large ABox?
Use OWL 2 EL
Use OWL 2 QL
Decidability is important?
Use any FOL, extension thereof, or higher order logic, e.g. Common Logic, DLRus
Use OWL (2) DL

Figure 5.4: A preliminary decision diagram to choose a suitable ontology language for one's prospective ontology (with indications of current typical usage and suggestions for use).

essential, and, e.g., graph-based versus model-theoretic semantics, and e.g., the positionalist or standard view commitments. This is interesting from a logic and philosophical perspective at a more advanced level of ontology engineering and research, which we will not cover in this introductory course to a practically usable detail.

Encoding Peculiarities

This is tricky to grasp at the start: there may be a difference between what the domain expert sees in the tool—what it is 'understood to represent'—and what you, as the computer scientist, know how it works regarding the computational representation at the back-end that a domain expert need not know about. For instance, a modeller sees an ontology in the interface of a software application, but it is actually stored in a database, or sees an n-ary relationship, but this is encoded as 3 binaries behind the scenes. The former has to do with the notion of representing classes as instances in the system (not classes-as-instances in the ontology!). For instance, Chair is a universal, class, or concept that is represented in the OWL ontology as class `Chair`, but one equally well can store Chair in a database table, by which it mathematically has become an instance, yet it is 'thought of' and pretended to be a universal, class, or concept in the graphical interface. This is primarily relevant for SKOS and OBO ontologies. Take the Gene Ontology, among others, which is downloadable in OBO or OWL format—i.e., its taxonomy consists of, mathematically, classes—and is available in database format—i.e., mathematically it is a taxonomy of instances. This does not have to be a concern of the subject domain experts, but it does affect how the ontology can be used in ontology-driven information systems. A motivation for storing the ontology in a database, is that databases are much better scalable, which is nice for querying large ontologies. The downside is that data in databases are much less usable for automated reasoning. As an ontology engineer, you will have to make such trade-offs.

There is no such choice for SKOS 'ontologies', because each SKOS concept is always serialised as an instance, as we shall see in Chapter 7.

One also could avail of "punning" as a way to handle second-order logic rules in a first-order setting and use the standard reasoners instead of developing a new one (that is, not in the mode of confusing class as instance, but for engineering reasons). This can be done by converting the TBox into an ABox, encode the second-order rules in the TBox and classify the classes-converted-into-individuals accordingly. We will come across one such example with OntoClean in the next chapter in Section 5.2.2.

In short: one has to be careful with the distinction between the 'intended meaning' and the actual encoding.

On formalizing it

The '*how to formalise it?*' question is not new at all, neither in IT and Computing [Halpin, 2001, Hofstede and Proper, 1998] nor in logic [Barwise and Etchemendy, 1993], and perhaps more of those advances made elsewhere should be incorporated in ontology development methodologies. Some preliminary recent efforts for micro-level ontology development that also includes usage of a foundational ontology are presented in [Keet, 2012b].

Step 7a in the above-mentioned outline for micro-level development concerns formalising the domain adhering to the modelling choices made, such as n-aries as classes or as object properties, how to handle the 'attributes', any approximations used in the less expressive language, typing of the object properties, and use of other modelling aides, such as OntoPartS, sample populations, and so forth. For instance, one can can formalise the statement that a person runs a marathon as two classes, Person and Marathon, and one object property, runs. We can't say that each person runs a marathon, but it is true that each marathon is run by at least one persons, so we could add to our ontology:

$\mathsf{Marathon} \sqsubseteq \exists\, \mathsf{runs}^{-}.\mathsf{Person}$

Another option is to make a whole class hierarchy of processes, including Running, add the two classes, and relate them through tow new object properties: a person participates in a Running process, and the Running process is part of a Marathon. The latter option may make more sense after going through the next chapter. Whichever way you choose, do try to do it consistently throughout.

5.2 Methods to improve an ontology's quality

The methodologies we have seen in the previous section may include methods at a particular stage. Methods that help the ontologist in certain tasks of the ontology engineering process include, but are not limited to, assisting the modelling itself, how to integrate ontologies, and supporting software tools. From a modeller's viewpoint, the comparatively 'easy' tool support is that of explanations and root justifications, which we only touch upon in the next section. The more challenging aspect is to model some piece of knowledge in a 'good' way, and have tool support to help you with it that is based on solid modelling principles, an example of which we shall look at afterward.

5.2.1 Explanation and justification

People make errors with respect to what they intend to represent in the ontology, or do it correctly, but are somewhat surprised by one or more deductions. The automated reasoners can help explain that, or: 'justify' the deduction. The more recent versions of ontology development environments have this feature already implemented, and you have come across it during the exercises (by having clicked on the "?" on the right of the yellow deduction in Protégé). Put differently: you have been using an automated reasoner to 'debug' the ontology.

The explanation feature uses the standard reasoning services and new reasoning services tailored to pinpointing the errors and explaining the entailments. Such 'debugging' goes under terms like glass box reasoning [Parsia et al., 2005], (root) justification [Horridge et al., 2008], explanation [Borgida et al., 2008], and pinpointing errors. While they are useful topics, we will spend comparatively little time on it, because it requires some more, and more in-depth, knowledge of Description Logics and its (mostly tableaux-based) reasoning algorithms. Those techniques use the automated reasoner to at least locate modelling issues—limited to logic alone, not ontological issues—and use additional algorithms to explain each deduction in the most succinct way, instead of just returning a bunch of inconsistent classes, near-'magical' reclassifications of classes, and several justifications. Proposing possible fixes automatically is yet a step further and work is under way to address that.

Despite brushing over the technical details, it may be useful anyhow to have a quick look at common mistakes and surprising consequences. Parsia et al. [Parsia et al., 2005] observed the following ones (in OWL DL ontologies) regarding unsatisfiable classes, undesirable inferred subsumptions, and inconsistent ontologies. The basic set of clashes for concepts (w.r.t. tableaux algorithms) are:

- Atomic: An individual belongs to a class and its complement
- Cardinality: An individual has a max cardinality restriction but is related to more distinct individuals
- Datatype: A literal value violates the (global or local) range restrictions on a datatype property

The basic set of clashes for knowledge bases (ontology + instances) are:

- Inconsistency of assertions about individuals, e.g., an individual is asserted to belong to disjoint classes or has a cardinality restriction but related to more individuals
- Individuals related to unsatisfiable classes
- Defects in class axioms involving nominals (`owl:oneOf`, if present in the language)

Knowing this should also help you with understanding the automatically generated explanations, especially when the ontology is large and the explanation as well, and, hence, figuring out a suitable repair for the defect.

More detailed information about typical misconceptions can also be found in, among others, [Rector et al., 2004] (who use the pizza ontology to describe them).

5.2.2 OntoClean to correct a taxonomy

OntoClean[4] [Guarino and Welty, 2009] helps the ontologist to find errors in a taxonomy, and explains why. One might ask oneself: who cares, after all we have the reasoner to classify our taxonomy anyway, right? Indeed, but that works only if you have declared many properties for the classes so that the reasoner can sort out the logical issues (recollect Section 3.3). However, it is not always the case that many property expressions have been declared and those reasoners does not detect any ontological issues.

OntoClean fills this gap for taxonomies. It uses several notions from philosophy, such as rigidity, identity criteria, and unity (based on [Guarino and Welty, 2000a] [Guarino and Welty, 2000b]) to provide modelling guidelines. Let's take rigidity as brief example (we go into the details during the lecture and tutorial). There are four different ones, but here we use just two: rigid and anti-rigid, defined as follows:

[4] `http://www.ontoclean.org/`

Definition 5.1 (+R). *A* rigid *property ϕ is a property that is essential to* all *its instances, i.e., $\forall x \phi(x) \rightarrow \Box\phi(x)$.*

Definition 5.2 ($\sim$R). *An* anti-rigid *property ϕ is a property that is not essential to* all *its instances, i.e., $\forall x \phi(x) \rightarrow \neg\Box\phi(x)$.*

We use definitions like these two to annotate each class in the ontology. For instance, a modeller may want to assert that Apple is rigid (each instance remains an apple during its entire existence) and being a Professor is anti-rigid (all individuals that are professors now were at some time not a professor).

Subsequently, we apply the meta-rules to reclassify the classes. For our rigid and anti-rigid meta-property, the applicable rule is as follows:

- *Given two properties, p and q, when q subsumes p the following constraint hold:*
    1. *If q is anti-rigid, then p must be anti-rigid*

Or, in shorthand: $+R \not\subset \sim R$, i.e., it cannot be the case that a class that is annotated as being rigid is subsumed by a class that is annotated as being anti-rigid.

For instance, if we have, say, both Student and Person in our ontology, then the former is subsumed by the latter, not vice versa, because Person is rigid and Student anti-rigid. If Person $\sqsubseteq$ Student were asserted, it would say that each person is a student, which we know not to be the case. Put differently, 1) it is not the case that all persons come into existence as students and die as students, and 2) it is not the case that if a student cease to be a student (e.g., graduates), then that object also ceases to be a person.

The machinery of OntoClean that we will cover in the lecture also provides us with the theory and machinery to solve the "red apple issue" we encountered in the first lecture: Apple is rigid (and a sortal), but its redness is not (see [Guarino and Welty, 2009] for a recent written account).

Besides manual analyses, currently, there are two approaches how to incorporate the ideas of OntoClean in OWL ontologies. One is to develop a separate application to handle the annotations of the classes and the rules, another is to leverage the capabilities of the standard reasoning services of the OWL reasoners, which is done by [Glimm et al., 2010, Welty, 2006]. Glimm et al.'s [Glimm et al., 2010] and Welty's [Welty, 2006] approach differ in detail, but they have in common the high-level approach:
1) develop the domain ontology (TBox);
2) push it into the ABox (i.e., convert everything from the TBox into ABox assertions);
3) encode the OntoClean 'meta rules' in the TBox;
4) run the standard OWL reasoner and classify the 'instances';
5) transfer the reclassifications in the taxonomy back into the domain-ontology-in-TBox.

The lecture will go into some detail of OntoClean (but not the details of the integration into OWL; you can choose to do so as a topic for a mini-project).

5.2.3 Detecting and revising modelling flaws

OntoClean does little to help solving so-called *undesirable deductions*, be they logically consistent or not, and the justifications computed may not always point to the root problem form a modelling viewpoint. Early works aiming at identifying typical modelling mistakes in OWL are described in [Rector et al., 2004], and, more recently, there are so-called "anti-patterns" of the 'don't do this' variety [Roussey et al., 2009], and a growing catalogue of pitfalls [Poveda-Villalón et al., 2012], of which 21 can be scanned automatically online with the OntOlogy Pitfall Scanner! (OOPS!)[5]. A recent evaluation of the presence of those 21 pitfalls showed that it does not make much difference whether the ontology is one developed by novices, an arbitrary ontology, or is a well-known ontology [Keet et al., 2013c][6]. It may well be that the notion of a good quality ontology is not tightly related to absence of pitfalls, or maybe the modelling pitfalls are propagated from the well-known ones by novice modellers; whichever be the case, it is fertile ground for research. Notwithstanding this, the ontology can be scanned quickly with OOPS! and the results provides pointers where the ontology may be improved.

The error, anti-pattern, and pitfall efforts look more at quality of ontology from the negative side—i.e., what are the mistakes?—whereas, e.g., OntoClean looks at the positive side (when is some representation good?). The *SubProS* and *ProChainS* compatibility services fall in the category of looking at the positive side (*SubProS* is an extension of the *RBox Compatibility service*, Section 6.2.2). They check for meaningful object property hierarchies and property chains. First, one has to know when object property expressions are good and safe (i.e., guaranteed not lead to an undesirable deduction), then test for violations of those principles, and finally have guidance on how a mistake can be revised, which is described in [Keet, 2012a].

A broader setting of ontology quality is described in [Vrandečić, 2009], which includes aspects such as accuracy, adaptability, clarity, completeness, computational efficiency, conciseness, consistency/coherence and organizational fitness, and it contains a review of domain and task-independent evaluation methods that cover, among others, syntax, semantics, representation, and context aspects.

Let us consider two examples of (typical) modelling flaws that you should try to avoid; more examples can be found in the literature referenced above.

Example 5.1. Say, you have to represent "a pizza Hawaii has as topping ham and pineapple"[7]. A modeller may be inclined to take the natural language description of the toppings quite literally, and add

$$\exists \mathsf{hasTopping}.(\mathsf{Ham} \sqcap \mathsf{Pineapple}) \tag{5.1}$$

However, this is not what the modeller really wants to say. The "$\sqcap$" means 'and', i.e., we have some intersection, and thus the "$(\mathsf{Ham} \sqcap \mathsf{Pineapple})$" is the OWL class with those objects that are *both* ham *and* pineapple. However, nothing is both, for meat and fruit are disjoint, so the pizza Hawaii in our ontology has a topping that is $\mathsf{Nothing}$. What we want to represent, is that from $\mathsf{PizzaHawaii}$ there are at least two outgoing relations for the toppings, being one to Ham and one to $\mathsf{Pineapple}$, i.e.,

$$\exists \mathsf{hasTopping}.\mathsf{Ham} \sqcap \exists \mathsf{hasTopping}.\mathsf{Pineapple} \tag{5.2}$$

In addition, one may want to add a so-called 'closure axiom' to say that all pizzas Hawaii "have as topping only ham and pineapple",

$$\forall \mathsf{hasTopping}.(\mathsf{Ham} \sqcup \mathsf{Pineapple}) \tag{5.3}$$

Note also here that there is not a one-to-one mapping between the imprecise natural language and the logic constructs: ham *and* pineapple, but using an 'or' $\sqcup$, which becomes clearer when we rephrase it as "all toppings are either ham or pineapple". ♢

[5]`http://www.oeg-upm.net/oops`

[6]Put differently: when you ant to justify some modelling decision, the "argument of authority", unfortunately, does not always hold

[7]and we ignore the fact that, according to Italians, pizzas are not supposed to have any fruit on a pizza—other than tomatoes—so the pizza Hawaii is not really an Italian pizza.

To the best of my knowledge, there is no tool yet that helps you sift through such issues. For the following example there is a, thus far, only manual procedure, but at least there is a clear set of tests to carry out.

Example 5.2. We are going to detect flaws in OWL property chains; the complete test to check for a safe chain is described in [Keet, 2012a]. Basically, the domain and range class from left to right in the chain on the left-hand side of the inclusion has to be equal or a superclass, and likewise for the outer domain (resp. range) on the left-hand side and range of the object property on the right-hand side of the inclusion. For instance, take the property chain $\mathsf{hasMainTable} \circ \mathsf{hasFeature} \sqsubseteq \mathsf{hasFeature}$ in the Data Mining and OPtimisation (DMOP) ontology, which is depicted in Figure 5.5. The two properties have domain and range axioms $\mathsf{hasMainTable} \sqsubseteq \mathsf{DataSet} \times \mathsf{DataTable}$ and $\mathsf{hasFeature} \sqsubseteq \mathsf{DataTable} \times \mathsf{Feature}$. The range of hasMainTable and domain of hasFeature match neatly, i.e., both are DataTable. However, the domain of hasMainTable is DataSet and the domain of hasFeature is DataTable, and DataSet and DataTable are non-disjoint sibling classes. The reasoner infers $\mathsf{DataSet} \sqsubseteq \mathsf{DataTable}$ because of the property chain, which is undesirable, because a set is not a subclass of a table. The *ProChainS* tests helps detecting such issues, and proposals how to revise such a flaw are also described in [Keet, 2012a]. Note that in this case, there was *not* a logical inconsistency according to the language and the automated reasoning services, but instead it was a modelling issue (more examples can be found in the paper). ◇

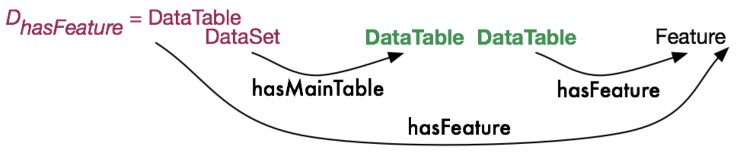


Figure 5.5: The property chain $\mathsf{hasMainTable} \circ \mathsf{hasFeature} \sqsubseteq \mathsf{hasFeature}$ with the domain and range axioms of the two object properties. (Source: [Keet, 2012a])

5.2.4 Tools

There are many tools around that help you with one method or with a methodology. Finding the right tool to solve the problem at hand (if it exists) is a skill of its own and it is a necessary one to find a feasible solution to the problem at hand. From a technologies viewpoint, the more you know about the goals, features, strengths, and weaknesses of available tools (and have the creativity to develop new ones, if needed), the higher the likelihood you bring a potential solution of a problem to successful completion.

Software-supported methodologies. WebODE[8] provides software support for Methontology, the NeOn toolkit[9] aims to support distributed development of ontologies, and likewise the MOdelling wiKI MOKI[10] (a Semantic Wiki[11]). The recent tools also reflect the afore-mentioned changes in typical ontology development settings, and it has been shown, for instance, that wiki-enhanced collaborative development is beneficial compared to not having such functionality [Franscescomarino et al., 2012].

[8]`http://webode.dia.fi.upm.es/`

[9]`http://neon-toolkit.org/wiki/Main_Page`

[10]`http://moki.fbk.eu/`

[11]For a range of semantic wiki projects, engines and features, check `http://semanticweb.org/wiki/Semantic_wiki_projects`

Ontology Development Environments (ODEs). Clearly, the tools listed under the 'Software-supported methodologies' are ODEs, but there are also ODEs that do not cater for a particular methodology. They mainly lack project management features, and/or the possibility to switch back and forth between informal, intermediate, and formal representation, or do not have features for project documentation. It may well be the case that such functionality is available in part or in whole as a set of plug-ins to the ODE. Some of those tools are stand-alone tools, such as Protégé and RacerPlus[12], others have a web interface for modeling and/or browsing, such as the DMOP-browser of the e-LICO project[13]. Most ODEs are packaged with one or more automated reasoners, but one also can use another one, given that there is a plethora of ontology reasoners and editors[14]. There are also various tool that have a pseudo-natural language interface or a graphical interface to adding axioms to an ontology.

Software-supported methods and other features. Most of the additional features and implemented methods exist as plugins for the ODEs or, thanks to the widespread uptake, have been integrated in the ODEs already upon installation. For instance, RacerPlus has extensive features for sophisticated querying, Protégé's PROMPT plugin[15] was one of the earlier tools for ontology integration, OWL ontology visualization with Ontograf is already included in the standard installation of Protégé, OntoClean with Protégé[16] (but recollect [Glimm et al., 2010, Welty, 2006]). There are many more other plug-ins for Protégé in its library[17], though before installing any of then, one has to verify the versioning of the plugins and the ODE.

Some recent tools that are focussed on improving the quality of the ontology are the Possible World Explorer[18], which helps with adding disjointness axioms [Ferré and Rudolph, 2012], and the OntOlogy Pitfall Scanner[19] (OOPS!) that implements an automated check of your ontology with 21 common modelling pitfalls [Poveda-Villalón et al., 2012].

Other features that can make an ontology developer's life easier, are the portals to search across ontologies, such as SWOOGLE[20] and the BioPortal[21], and those that also analyse the characteristics of the ontology, such as the TONES Ontology Repository.

Then there are tools for interaction with 'peripheral' usage of ontologies, most notably a conversion from OWL to latex so as to obtain the—to some, more readable—DL notation of the ontology (see "save as" in Protégé, select latex), and to automatically generate documentation alike software documentation—to others, more readable—like in LiveOWL [Peroni et al., 2012], and natural language verbalization (such as ACE, whig will be discussed in Chapter 7.)

For much longer listings of tools, see the list of semantic web development tools[22], and an updated list[23].

[12]`http://www.racer-systems.com/products/plus.phtml`
[13]`http://www.e-lico.eu`
[14]`http://www.w3.org/2007/OWL/wiki/Implementations`
[15]`http://protege.stanford.edu/plugins/prompt/prompt.html`
[16]`http://protege.stanford.edu/ontologies/ontoClean/ontoCleanOntology.html`
[17]`http://protegewiki.stanford.edu/index.php/Protege_Plugin_Library`
[18]`http://www.irisa.fr/LIS/softwares/pew`
[19]`http://oeg-lia3.dia.fi.upm.es/oops/index.jsp`
[20]`http://swoogle.umbc.edu/`
[21]`http://bioportal.bioontology.org`
[22]`http://esw.w3.org/topic/SemanticWebTools`
[23]`http://www.w3.org/2001/sw/wiki/Tools`

Developing your own tool. Use the OWL API or Jena toolkit. More TBA.

5.3 Design parameters and their dependencies

Ontology development methodologies provide one or more scenarios, but they do not address the dependencies between the permutations at the different stages in the development process. As a first step towards methodologies that gives a general scope, we will look at a range of parameters that affect ontology development in one way or another, and which, thus, should appear somewhere in the methodology deployed when creating an ontology. Analyzing the dependencies between helps one to answer questions such as:

1. If the purpose of my ontology is to access lots of data, then does that constrain the language I can use, and if so, how?
2. What reasoning services do I have if I choose as language OWL DL?
3. I have this ontology in OWL 2DL that I want to import into mine that I will use for ontology-driven natural language processing; can I reuse that ontology as is?

The parameters considered are: the typical main purpose(s) that the ontology is to serve, reuse of ontologies, what your 'starting material' was (i.e., a notion of bottom-up ontology development), the ontology languages we have seen in the previous chapters, and reasoning services, including those we used in the previous chapter. They are described in more detail in [Keet, 2010a] and summarised in the first column and row in Figure 5.6. All dependencies between these parameters have been assessed and useful combinations are motivated [Keet, 2010a][24]; the table with the outcome is included here as Figure 5.6, which was assessed against a set of ontologies and a survey among ontology developers, whose results concur with the theoretical assessment.

You will gain practice with it in the exercises. As for the three questions in the text, what do you think? 1) Indeed, it does constrain your language. 2) Can you recall from the previous chapter what the so-called "standard reasoning services" are? In addition, the table has a "+" sign for non-standard and ontological reasoning services; we will see those in the next section. 3) no. The dependencies are due to, primarily, computational challenges and types and subject domain of the ontologies. For instance, if one wants to query lots of data by means of an ontology (or: have a large ABox), then the ontology language has to be scalable and have a low data complexity to keep it tractable (recall Section 4.3); practically, this limits one to using OWL 2 QL as ontology language. In addition, most foundational ontologies (which we'll discuss in detail in the next chapter) use a rather expressive ontology language so as to be as precise as possible, which is then in conflict with the scalability requirement. But a comprehensive foundational ontology fits well when one wants to develop an ontology to represent a scientific theory (assuming the ontology does not become too large).

Once we have covered more about ontology development, you may want to have a look at the table again, or even add more parameters and dependencies.

5.4 Exercises

Exercise 22. Figure 5.6 summarises dependencies between ontology development parameters, which are discussed in more detail in [Keet, 2010a]. Discuss the feasibility of the following combinations, and make an informed guess about the unknowns:

[24]for a shorter and lighter version applied to agri-ontologies, see [Keet, 2009c].

		OWL Language					Ontology reuse		
	SKOS	2 QL	2 EL	2 DL	DL	Extensions	foundational	reference	domain
Purpose ⇓									
1. Query data	–	+	–	–	–	+	–	–	±
2. Database integration	+	+	+	–	–	±	±	±	+
3. Integration / record navigation	+	+	+	–	–	–	–	±	+
4. Part of scientific discourse	–	–	–	+	+	+	+	+	+
5. Web services orchestration	–	–	+	±	+	–	±	+	+
6. ODIS	±	+	+	–	±	±	±	+	+
7. ontoNLP	+	+	+	–	±	–	±	+	+
8. Science	–	–	–	+	±	+	+	+	–
9. Tutorial ontology	–	–	–	+	+	±	–	–	+
Reasoning services ⇓									
1. Standard	–	±	±	+	+	+			
2. Non-standard	–	±	±	+	+	–			
3. Querying	–	+	+	–	–	±			
4. Ontological	+	+	+	+	+	+			
Bottom-up ⇓									
1. Other KR/CM	–	±	±	+	+	–			
2. DB reverse	–	±	±	+	+	–			
3. Textbook models	–	–	±	+	+	+			
4. Thesauri	+	±	+	–	–	–			
5. Other semi-structured	±	±	+	–	–	–			
6. Text mining	+	±	+	–	–	–			
7. Terminologies	+	±	+	–	–	–			
8. Tagging	+	±	+	–	–	–			
Ontology reuse ⇓									
1. Foundational	–	–	–	+	±	–			
2. Reference	–	±	±	+	+	–			
3. Domain	±	±	+	+	+	–			

Figure 5.6: Basic cross-matching between realistic combinations of parameters. The more complex dependencies, such as the interaction between purpose, language, and reasoning service, can be obtained from traversing the table (*purpose* ↔ *language* and *language* ↔ *reasoning services*), likewise for, e.g., the dependencies between purpose and bottom-up development, assessing *purpose* ↔ *language* and then *bottom-up* ↔ *language*. '–' indicates 'discouraged or not possible', '±' a 'might work', and '+' as a 'workable or good combination'. (Source: [Keet, 2010a])

a. Purpose: science; Language: OWL 2 DL, or an extension thereof; Reuse: foundational; Bottom up: form textbook models; Reasoning services: standard and non-standard.
b. Purpose: querying data through an ontology; Language: some OWL 2; Reuse: reference; Bottom up: physical database schemas and tagging; Reasoning services: ontological and querying.
c. Purpose: ontoNLP; Language: OWL 2 EL; Reuse: unknown; Bottom up: a thesaurus and tagging experiments; Reasoning services: mainly just querying.
d. Apply it to the ontology you are developing for the Practical Assignment (see Appendix A).

Exercise 23. Ontology development methodologies have evolved over the past 20 years. Compare the older METHONTOLOGY with the newer NeON methodology.

Exercise 24. Describe and apply the OntoClean *rules* to the flawed ontology depicted in Figure 5.7, i.e., try to arrive at a 'cleaned up' version of the taxonomy by using the rules.

Exercise 25. Take the Pizza ontology, and submit it to the OOPS! portal. Based on its output, what would you change in the ontology, if anything?

Exercise 26. Work on your own ontology for the Practical Assignment (see Appendix A).

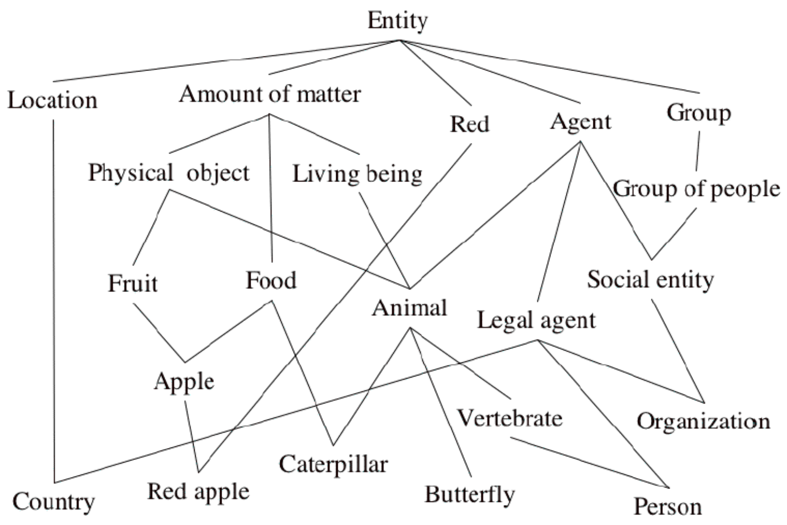


Figure 5.7: An ‘unclean’ taxonomy. (Source: OntoClean teaching material by Guarino)

5.5 Literature and reference material

1. Guarino, N. and Welty, C. An Overview of OntoClean. in S. Staab, R. Studer (eds.), *Handbook on Ontologies*, Springer Verlag 2004, pp. 151-172.
2. Keet, C.M. Dependencies between Ontology Design Parameters. *International Journal of Metadata, Semantics and Ontologies*. 2010, 5(4): 265-284.
3. Mari Carmen Suarez-Figueroa, Guadalupe Aguado de Cea, Carlos Buil, Klaas Dellschaft, Mariano Fernandez-Lopez, Andres Garcia, Asuncion Gomez-Perez, German Herrero, Elena Montiel-Ponsoda, Marta Sabou, Boris Villazon-Terrazas, and Zheng Yufei. NeOn Methodology for Building Contextualized Ontology Networks. NeOn Deliverable D5.4.1. 2008.
4. Alexander Garcia, Kieran ONeill, Leyla Jael Garcia, Phillip Lord, Robert Stevens, Oscar Corcho, and Frank Gibson. Developing ontologies within decentralized settings. In H. Chen et al., editors, *Semantic e-Science. Annals of Information Systems 11*, pages 99-139. Springer, 2010.
5. Gomez-Perez, A., Fernandez-Lopez, M., Corcho, O. *Ontological Engineering.* Springer Verlag London Ltd. 2004.

CHAPTER 6

Top-down Ontology Development

Having an ontology language is one thing, but *what* to represent, and *how*, is quite another. In the previous chapter, we looked at answering "Where do you start?" with methodologies, but, to some extent, we are still left with answering: How can you avoid reinventing the wheel? What can guide you to make the process easier to carry it out successfully? How can you make the best of 'legacy' material? There are two principal approaches, being the so-called *top-down* and *bottom-up* ontology development with their own set of methods, tools, and artefacts; in this chapter, we focus on the former and in the next chapter on the latter, where each can be seen as a refinement of some aspects of an overall methodology like introduced in Chapter 5. In this chapter we look at 'avoiding to reinvent the wheel' and 'what can guide you to make the process of adding those axioms easier' by reusing some generic principles. Those generic modelling aspects are typically represented in *foundational ontologies* and, to a lesser extent, *ontology design patterns*.

6.1 Foundational ontologies

The basic starting point for top-down ontology development is to consider several core principles of Ontology for ontologies; or: some philosophical guidance for the prospective engineering artefact[1]. Although we will not enter in deep debates about philosophical theories in this course, it is useful to know it has something to offer to the development of ontologies, and we will see several examples where it has had influence.

A few of such examples where results from philosophy can be useful in deciding what is going to be represented in your ontology, and, to some extent, how, are as follows. One can commit to a 3-Dimensional view of the world with objects persisting in time or,

[1]As philosophy enters, a note about terminology may be in order, because some ideas are borrowed and changed, and some terms that are the same do mean different things in different disciplines. In the literature, you will come across *material* ontology and *formal* ontology. The former (roughly) concerns making an 'inventory' of the things in the universe (we have the vase, the clay, the apple, etc.), whereas the latter concerns laying bare the formal structure of (and relation between) entities, which are assumed to have general features and obey some general laws that hold across subject domains, like identity, constitution, and parthood (the latter will be introduced in Section 6.2). So, in ontology engineering the 'formal' may refer to *logic-based* but also to the usage of 'formal' in philosophy, which concerns the topic of investigation and does not imply there is a formalisation of it in a logic language. In most computer science and IT literature, when 'formal' is written, it generally refers to logic-based.

instead, a perdurantist one (4-Dimensional) with space-time worms; e.g., are you convinced that you after reading this sentence is a different you than you before reading this sentence? If so, then you may well be a perdurantist, if you consider yourself to be the very same entity before and after, then you lean toward the 3D, endurantist, commitment (but before proclaiming to be one or the other based on this single example, do read up on the details and the implications). Other philosophical distinctions concern whether you are concerned with (in OWL terminology) classes or individuals, with universals and/or concepts, and whether your ontology intended to be descriptive or prescriptive. Then there more detailed decision to make, such as whether you are convinced that there are entities that are not in space/time (i.e., that are abstract), whether two entities can be co-located (the vase and the amount of clay it is made of), what it means that one entity is [dependent on/constituted by/part of/...] another? And many more of such questions and decision to make. Fortunately, if you do not want to entertain yourself with these questions, you can take someone else's design decisions and use that in ontology development. Someone else's design decision for a set of such questions typically is available in a **foundational ontology**, and the different answers to such questions end up as different foundational ontologies (even with the same answers they may be different; see, e.g. beyond concepts [Smith, 2004], the WonderWeb deliverable [Masolo et al., 2003], and a synopsis of the main design decisions for DOLCE [Borgo and Masolo, 2009]). The intricacies of, and philosophical debates about, the more subtle details and differences are left to another course, as here the focus is one why to use one, where, and how; differences will be discussed in the lecture insofar as they affect foundational ontology choice and usage.

6.1.1 Typical content of a foundational ontology

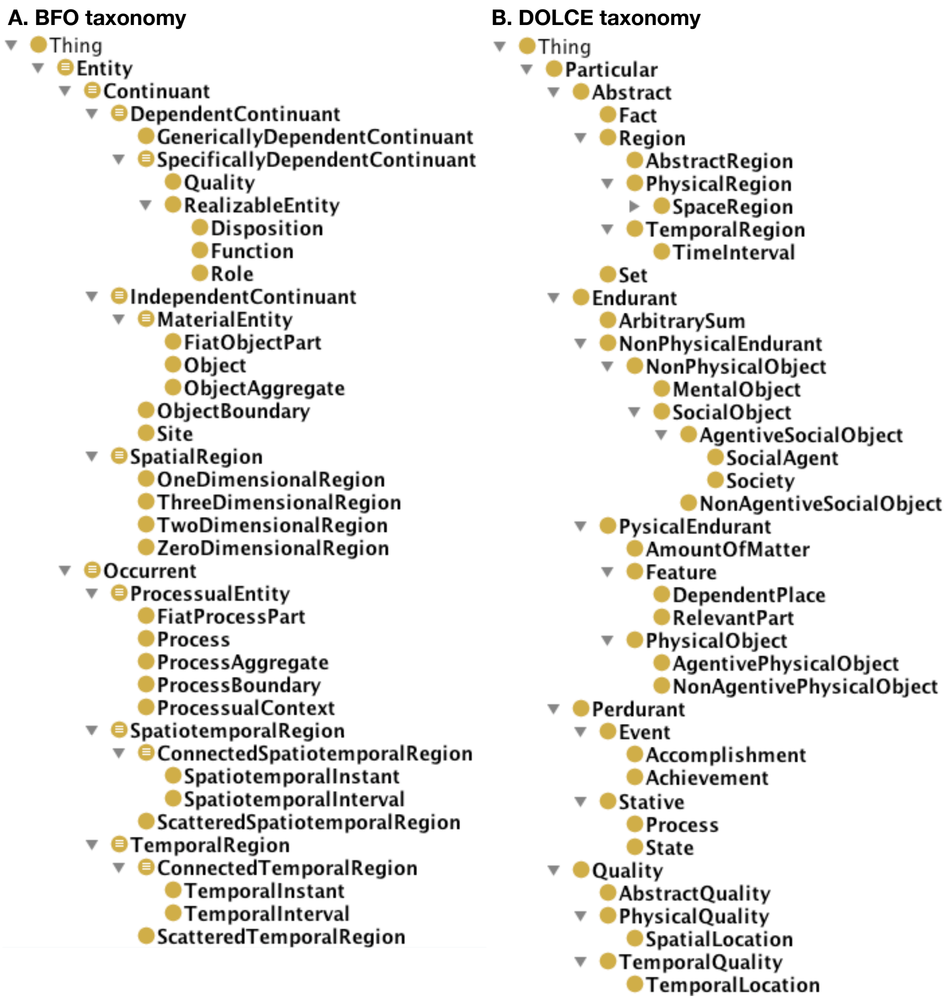
Foundational ontologies provide a high-level categorization about the kinds of things you will represent in the ontology, such as *process* and *physical-object*, relations that are useful across subject domains, such as *participates-in* and *part-of*, and (what are and) how to represent 'attributes' such as Colour and Height (recall Section 1.3), e.g., as *qualities* or some kind of *dependent continuant* or *trope*. To make sense of this, let us start with the two main ingredients: the 'class' taxonomy and the relationships.

Universals, categories, class hierarchy

Just like with other ontologies we have seen, also a foundational ontology represented in OWL has a hierarchy in the 'TBox'. However, there are some differences with that of a domain ontology such as the Pizza ontology. The hierarchy in a foundational ontology does not contain subject domain classes such as Boerewors and PizzaHawaii, but *categories* (or, loosely, 'conceptual containers') of kinds of things. For instance, all instances of PizzaHawaii can be considered to be *physical objects*, and are those sausages that are an instance of Boerewors. If we assume there to be physical objects, then presumably, there can also be entities that can be categorised as *non-physical objects*. Organisations, such as UCT, fall in the category of *social object*, which are a type of non-physical object. Likewise, one can categorise kinds of processes. For instance, 'writing an exam' is something that unfolds in time and has various sub-activities, such as thinking, writing, erasing pencil marks, and so on; taken together, writing an exam is an *accomplishment*. Contrast this with Sitting: for the whole duration you sit, each part of it is still an instance of sitting, which thereby can be categorised as a *state*. None of the things mentioned in italics in this paragraph actually are specific entity types that you would encounter in an ontology about subject domain entities only, yet we would want to be able to categorise the kinds of things we represent in our domain ontology (the 'why' is explained further below). It is these and other categories that are represented in a foundational ontology[2].

The categories introduced above actually are from the the Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE) foundational ontology, and a screenshot of its hierarchy is shown in Figure 6.1-B; behind this simple taxonomy is a comprehensive formalisation in first order predicate logic that was introduced in [Masolo et al., 2003].

Being a pedantic ontologist, one could go as far as saying that if a category is not in the foundational ontology, then its developers are of the opinion it does not exist in reality.



A. BFO taxonomy

- Thing
  - Entity
    - Continuant
      - DependentContinuant
        - GenericallyDependentContinuant
        - SpecificallyDependentContinuant
          - Quality
          - RealizableEntity
            - Disposition
            - Function
            - Role
      - IndependentContinuant
        - MaterialEntity
          - FiatObjectPart
          - Object
          - ObjectAggregate
        - ObjectBoundary
        - Site
      - SpatialRegion
        - OneDimensionalRegion
        - ThreeDimensionalRegion
        - TwoDimensionalRegion
        - ZeroDimensionalRegion
    - Occurrent
      - ProcessualEntity
        - FiatProcessPart
        - Process
        - ProcessAggregate
        - ProcessBoundary
        - ProcessualContext
      - SpatiotemporalRegion
        - ConnectedSpatiotemporalRegion
          - SpatiotemporalInstant
          - SpatiotemporalInterval
        - ScatteredSpatiotemporalRegion
      - TemporalRegion
        - ConnectedTemporalRegion
          - TemporalInstant
          - TemporalInterval
        - ScatteredTemporalRegion

B. DOLCE taxonomy

- Thing
  - Particular
    - Abstract
      - Fact
      - Region
        - AbstractRegion
        - PhysicalRegion
          - SpaceRegion
          - TemporalRegion
            - TimeInterval
      - Set
    - Endurant
      - ArbitrarySum
      - NonPhysicalEndurant
        - NonPhysicalObject
          - MentalObject
          - SocialObject
            - AgentiveSocialObject
              - SocialAgent
              - Society
            - NonAgentiveSocialObject
      - PysicalEndurant
        - AmountOfMatter
        - Feature
          - DependentPlace
          - RelevantPart
        - PhysicalObject
          - AgentivePhysicalObject
          - NonAgentivePhysicalObject
    - Perdurant
      - Event
        - Accomplishment
        - Achievement
      - Stative
        - Process
        - State
    - Quality
      - AbstractQuality
      - PhysicalQuality
        - SpatialLocation
      - TemporalQuality
        - TemporalLocation

Figure 6.1: Screenshots of the OWLized BFO and DOLCE taxonomies; for indicative purpose: Perdurant ≈ Occurrent, Endurant ≈ IndependentContinuant.

Relations in foundational ontologies

In analogy to the 'subject domain classes' in domain ontologies versus categories in foundational ontologies, one can identify generic relations/relationships/object properties that are different from those in domain ontologies. For instance, a domain ontology about

[2]Some call such categories universals, but that debate is beyond the scope of the course.

universities may have a relation enrolled to relate Student to Course, or in a sports ontology that a runner runs a marathon. These relations are specific to the subject domain, but there are several that re-appear across domains, or: they are subject domain-independent. Such subject domain-independent relations are represented in a foundational ontology. Notable relations are *part-of* (which we shall look at in some detail in Section 6.2), that an object *participates-in* an event, that some object (e.g., a vase) is *constituted-of* an amount of matter (such as clay), and *dependency*, for when the existence of one entity depends on the existence of another. The characterisation of such relations go hand in hand with the categories, for we can assert that, say, *participates-in* holds only between an *endurant* (an entity that is more general than object, wholly present at a time) and a *perdurant* (an entity that is more general than event; those things that unfold in time).

For the sake of example of foundational ontologies as well as visualizations of ontologies, a screenshot of the top-level entities and relationships is shown in Figure 6.2: the dashed (non-purple) lines in Figure 6.2 graphically represent such relations between the entities.

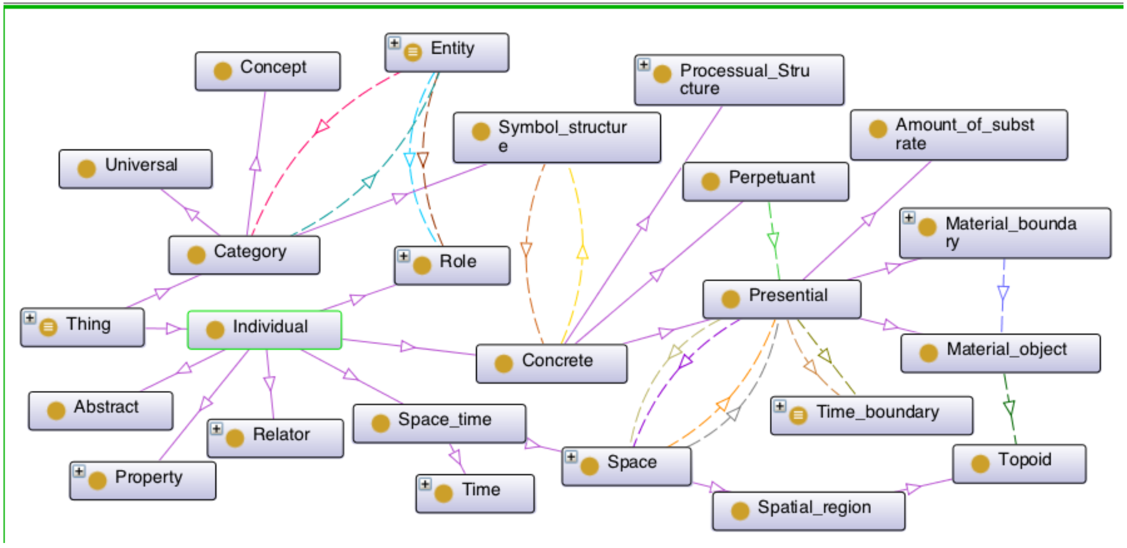


Figure 6.2: Screenshots of the Ontograf rendering of the top-level categories and their relations of the GFO (solid lines: subsumption; dashed lines: various object properties).

Attributions

The third main component of representing knowledge are attributions, as, just like in conceptual data modelling, 'attributes' have to be represented somehow. There is a domain-specific component to it and there are general, recurring, principles of attributions, and it is the latter than are captured in an foundational ontology—to some extent at least. However, this is done quite differently from attributes you have modelled in UML Class Diagrams.

Let us first revisit the 'attribute' Colour we have seen in Section 1.3. We could decide to make it a data property in OWL, declare its domain to be Rose and choose the data type `String`, i.e., hasColour $\mapsto$ Rose$\times$`String` in ontology $\mathcal{O}_1$, or, in OWL functional syntax style notation:

```
DataPropertyDomain(ex:hasColour ex:Rose)
DataPropertyRange(ex:hasColour xsd:string)
```

That is, a binary relationship, which is the same approach as in UML class diagrams.

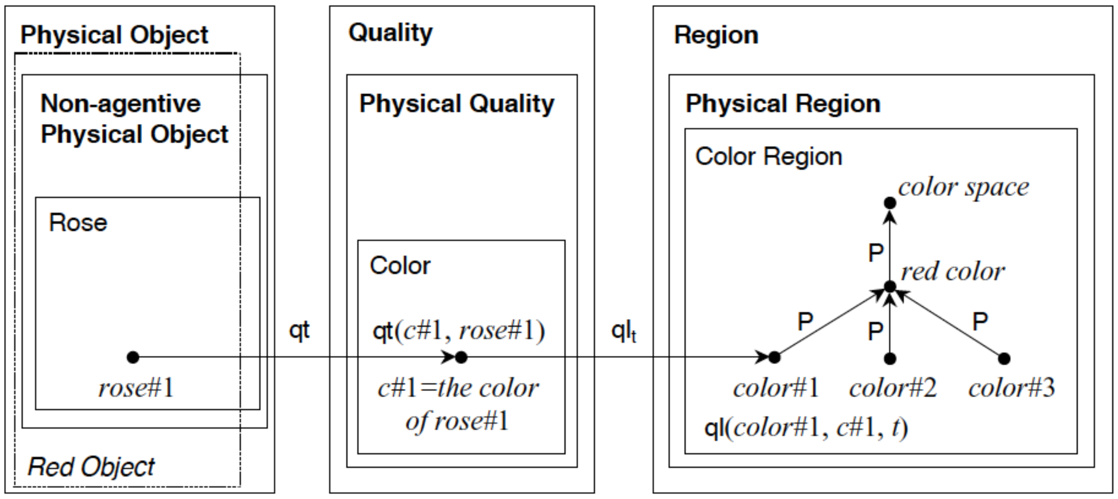


Figure 6.3: DOLCE's approach for qualities ('attributes') (Source: [Masolo et al., 2003])

If another ontology developer decides to record the values in integers in $\mathcal{O}_2$, then the hasColour properties in $\mathcal{O}_1$ and $\mathcal{O}_2$ are incompatible in the representation, albeit not in 'conceptualization', because the notion of colour is the same throughout. Another option, which is rather typical of foundational ontologies and their OWL-ized version, is to represent such 'attributes' as unaries somewhere in the class hierarchy of the OWL ontology, such as a subclass of Quality in DOLCE[3] [Masolo et al., 2003] or as (specifically) dependent continuant in the Basic Formal Ontology (BFO[4]). An example of the approach taken in DOLCE is depicted in Figure 6.3: rose1 is an instance of Rose, which is a subclass of Non-Agentive Physical Object, and it is related by the qt relation to its colour property, c1, which is an instance of the quality Colour that is a subclass of Physical Quality. The actual value—the [measured] redness—of the colour of the rose at a given time is a region red colour as instance of the Colour Region, which is a subclass of Physical Region, and they are related by means of the $\mathsf{ql_t}$ relation.

You may wonder what is the point of fussing about this distinction, let alone choosing the more cumbersome second option. To go into the argument, recall first that an important purpose of ontologies is system interoperability and integration (see Figure 1.4). Consider the scenario where ontology $\mathcal{O}_1$ = `AWO.owl` has a data property hasWeight for any object, but then with its XML data type set to `integer`. One can declare, e.g., Elephant $\sqsubseteq$ =1 hasWeight.`integer`. Perhaps a hasWeightPrecise with as data type `real` may be needed elsewhere; e.g., in ontology $\mathcal{O}_2$ about animals in the zoo. Implicitly, it was assumed by the developers that the weight would be measured in kg. Now someone comes along and wants to use the ontology, but wants to record the weight in lbs, which then amounts to adding, say, hasWeightImperial, and so on, all about weights. Put differently, what happens is a replication of the very same issues encountered in database integration. But this was precisely what ontologies were supposed to solve! Copying the problem from conceptual data modelling into the ontologies arena, just because you're more familiar with that way of modelling things, is not going to solve the interoperability problem. Thus, we need something else than sneaking application decisions about how to store the data into the ontology. The idea is to *generalise* (more precisely: reify) the attribute into a class so that we can reuse the core notion that is the same throughout (Weight in our example), and this new entity is then related to the endurants and perdu-

[3]`http://www.loa-cnr.it/DOLCE.html`
[4]`http://www.ifomis.org/bfo`

rants on the one side and instead of datatypes, we use value regions. Thus, an unfolding from one attribute/OWL data property into at least two properties: there is one OWL object property from the endurant/perdurant to the reified attribute—a quality property, represented as an OWL class—and a second object property to the value(s). In this way, the shared understanding can be shared, and any specifics on how one has to store the data is relegated to the implementation, therewith solving the problem of the limited reusability of attributes and preventing duplication of data properties.

The only remaining step is that the foundational ontologies are silent about the case when one really has to represent some values in the ontology itself. One option is presented in [Keet et al., 2013b] for the data mining optimization ontology: from the Region class, a data property hasDataValue was added with as XML data type `anyType`. This allows one to use the attributions across ontologies and data mining tools, yet leaves the flexibility to the implementer to choose the data type.

This concludes the brief idea of what is in a foundation ontology. As you may have observed from Figures 6.1 and 6.2, there are several foundational ontologies, which may be confusing or looking like me overcomplicating things, so we spend a few words on that now.

On multiple foundational ontologies

You may wonder what else is contained in DOLCE, and whether there are other foundational ontologies besides DOLCE, which can be answered in the affirmative; e.g., General Formal Ontology (GFO[5]) [Herre and Heller, 2006], natural language focused GUM[6], and SUMO[7] [Nikitin et al., 2003]. The top-level taxonomic structure of DOLCE and BFO is depicted in Figure 6.1, but note that DOLCE (as well as the GFO) has a rich axiomatisation with plenty of object properties, which at some point clutters the graphics, as can be seen for a section of the GFO in Figure 6.2. The respective documentation has further details about their formalisation and the rationale for having modelled it in the way they did. For instance, DOLCE takes a multiplicative approach, GFO lets you represent both universals and individuals in the same ontology, BFO claims a realist approach, and so on. Their properties have be structured and are included in the ONSET tool[8] that helps you selecting one for one's own domain ontology based on the requirements the user selects [Khan and Keet, 2012]. You will experiment with this during the exercises in the lab: it saves you reading the foundational ontology literature to large extent, and all of those new terms that have been introduced (like "multiplicative") have brief informal explanations.

One can wonder whether such foundational ontologies just use different names for the same kind of entities, but are essentially all the same anyway. Only very few detailed comparisons have been made. If we ignore some intricate philosophical aspects, such as whether universals and properties exist or not, then only few entity-by-entity alignments can be made, and even less mappings. An alignment is a mapping only if asserting the alignment in the new ontology containing the (foundational) ontologies does not lead to an inconsistency. Table 6.1 lists the common alignments among DOLCE, BFO, and GFO. More alignments and mappings are described and discussed in [Khan and Keet, 2013a, Khan and Keet, 2013c] and a searchable version is online in the foundational ontology

[5]`http://www.onto-med.de/ontologies/gfo/`
[6]`http://www.fb10.uni-bremen.de/anglistik/langpro/webspace/jb/gum/index.htm`
[7]`http://www.ontologyportal.org/`
[8]`http://www.meteck.org/files/onset/`

library ROMULUS[9] [Khan and Keet, 2013b].

Table 6.1: Common alignments between DOLCE-Lite, BFO and GFO; the ones numbered in bold can also be mapped. (Source: [Khan and Keet, 2013a])

	DOLCE-Lite	**BFORO**	**GFO**
		Class	
1.	endurant	Independent Continuant	Presential
2.	physical-object	Object	Material_object
3.	perdurant	Occurrent	Occurrent
4.	process	Process	Process
5.	quality	Quality	Property
6.	space-region	SpatialRegion	Spatial_region
7.	temporal-region	Temporal-Region	Temporal_region
		Relational property	
1.	proper-part	has_proper_part	has_proper_part
2.	proper-part-of	proper_part_of	proper_part_of
3.	participant	has_participant	has_participant
4.	participant-in	participates_in	participates_in
5.	generic-location	located_in	occupies
6.	generic-location-of	location_of	occupied_by

In closing, note that there are different versions of each foundational ontology, not only differentiating between a formalisation on paper versus what is representable in OWL, but also more and less detailed version of an ontology. In addition, while DOLCE contains several relations it deems necessary for a foundational ontology, BFO-in-OWL has only a taxonomy, a separate theory of parthood relations, and an extension including the Relation Ontology. The Relation Ontology [Smith et al., 2005] was developed to assist ontology developers in avoiding errors in modelling and assist users in using the ontology for annotations, and such that several ontologies would use the same set of agreed-upon defined relations to foster interoperability among the ontologies. Philosophically, it is still a debate what then the 'essential' relations are to represent reality, and if those included are good enough, are too many, or too few. Currently, several extensions to the RO are under consideration[10] and refinements have been proposed, such as for RO's *transformation_of* [Keet, 2009a] that avails of theory underlying OntoClean (that we address in a later lecture) and the *derived_from* relation [Brochhausen, 2006].

The lecture will go into some detail of the foundational ontologies.

6.1.2 Using a foundational ontology

Having some idea of what a foundational ontology is, is one thing, but how to use them is a different story, and one that is not fully resolved yet. In this subsection, we start first with answering why one would want to use one at all, and some examples where it helps a modeller in making modelling decisions for the overall (domain) ontology. We then turn to some practical aspects, such as their files, language used, and how (where) to link one's domain entities to those generic categories in a foundational ontology.

[9] `http://www.thezfiles.co.za/ROMULUS/home.html`

[10] `http://www.bioontology.org/wiki/index.php/RO:Main_Page`

Why use a foundational ontology?

Foundational ontologies exist, but does that means one necessarily must use one? Not everybody agrees on the answer. There are advantages and disadvantages to it. The principal reasons for why it is beneficial are:

- one does not have to 'reinvent the wheel' with respect to the basic categories and relations to represent the subject domain,
- it improves overall quality, and
- it facilitates interoperability among ontologies.

In addition, a foundational ontology is useful when subtle distinctions, recognizing disagreement, rigorous referential semantics, general abstractions, careful explanation and justification of ontological commitment, and mutual understanding are important[11].

A subset of domain ontology developers do not see a benefit:

- they consider them too abstract, too expressive and comprehensive for the envisioned ontology-driven information system, and
- it takes excessive effort to understand them in sufficient detail such that it would not weigh up to the benefits.

A controlled experiment has been carried out with 52 novice ontology developers, which showed that, on average, using a foundational ontology resulted in an ontology with more new classes and class axioms, and significantly less new object properties than those who did not, there were no part-of vs. is-a mistakes, and, overall, "the 'cost' incurred spending time getting acquainted with a foundational ontology compared to starting from scratch was more than made up for in size, understandability, and interoperability already within the limited time frame of the experiment" [Keet, 2011b]. There is room for further experimentation, but results thus far point clearly to a benefit.

Modelling guidance: examples of some principal choices

An immediate practical benefit is that Ontology and foundational ontologies help preventing making novice ontology developer's mistakes, such as confusing parthood with subsumption and class vs instance mix-ups. The former will become clear in the next lecture and in Section 6.2 (e.g., a province is part of a country, not a subclass). Regarding the latter, ontologically, instances/individuals/particulars are, roughly, those things that cannot be instantiated, whereas classes (or universals or concepts) can. For instance, the chair you are sitting on is an instance whereas the class Chair can be instantiated (the one you are sitting on is one such instance). Likewise, MacBookPro is a type of laptop, which in an OWL ontology would be added as a *subclass* of Laptop, not as an instance of Laptop—the MacBook I have with serial number ♯123456 is an instance, and, likewise, GoldenDelicious is a subclass of Apple, not an instance (the actual instances grow on the tree and are on the shelves in the supermarket).

The example on choosing how to represent relations is described in the next example

Example 6.1. A relation, i.e., an n-ary with $n > 1$, can be represented as an unary entity (a class in OWL) or as a n-ary relation (object property in OWL). It is certainly more intuitive to keep the n-aries as such, because it indicates a close correspondence with natural language. For instance, in formalising "Person runs marathon", it is tempting to represent the "runs" as an object property runs.

The foundational ontologies take a different approach. Such perdurants, like running, and the verbs we use to label them, are included as an *unary* (OWL class) suitably

[11] `http://ontolog.cim3.net/file/resource/presentation/NicolaGuarino_20060202/DOLCE--NicolaGuarino_20060202.pdf`

positioned as a subclass of 'processes', being Perdurant in DOLCE and Occurrent in BFO, which are then related with a *new* relation to 'objects', which are suitably positioned subclasses of Endurant in DOLCE (Continuant in BFO), in such a way that an endurant is a participant in a perdurant. For instance, still with the TBox-level knowledge that "Person runs marathon", then Running (being a subclass of Process) has_participant some Person and another binary to Marathon, but there is no 1-to-1 formalisation with an object property runs that has as domain and range Person and Marathon.

The latter results in a more compact representation, is intuitively closer to the domain expert's understanding, and makes it easier to verbalise the ontology, and therefore is likely to be more useful in praxis. The former is more generic, and thereby likely to increase reusability of the ontology. No scientific experiments have been conducted to test which way would be 'better' to represent such knowledge, and current mapping tools do not deal with such differences of representing roughly the same knowledge in syntactically very different ways. Whichever way you choose, sticking to that choice throughout the ontology makes the ontology easier to process and easier to understand by the human reader. $\diamondsuit$

A longer and practical example with the African Wildlife Ontology is included in the next section.

Practical aspects on using a foundational ontology

It was already mentioned that there are OWL-ised versions of several foundational ontologies, but there is more to it. Once the most appropriate foundational ontology is selected, the right version needs to be imported either in full or a module thereof, and it has to be linked to the entities in your ontology. The latter means you will have to find out which category each of your entity is and which object properties to use.

Within the Wonderweb Project[12], the participants realized it might not be feasible to have one singe foundational ontology that pleases everybody; hence, the idea emerged to create a library of foundational ontologies with appropriate mappings between them so that each modeller can choose her pet ontology and the system will sort out the rest regarding the interoperability of ontologies that use different foundational ontologies. The basis for this has been laid with the Wonderweb deliverable D18[13], but an implementation was yet to be done and new foundational ontology developments have taken place since 2003. A first step in the direction of such a foundational ontology library has been laid recently with the Repository of Ontology for MULtiple USes, ROMULUS [Khan and Keet, 2013b]. ROMULUS focuses on OWL ontologies in particular, but one should not forget that DOLCE also has a more comprehensive paper-based formalisation in a first order predicate logic, and that BFO also has a version of it in first order logic in the Isabelle theorem prover syntax.

The leaner OWL versions of DOLCE and BFO have been made available and are intended to be used for development of ontologies in one's domain of interest. These files can be found on their respective websites at the LOA[14] and IFOMIS[15] (which also lists domain ontologies that use them). Observe that DOLCE-Lite is encoded in the DL language that is characterized by $\mathcal{SHI}$, BFO is simpler (in $\mathcal{ALC}$); that is, neither one uses all OWL-DL capabilities of $\mathcal{SHOIN}(\mathcal{D})$, let alone of OWL 2 DL. Another difference is that BFO-in-owl is only a bare taxonomy (an extension with the Relation

[12]`http://wonderweb.semanticweb.org/`
[13]`http://wonderweb.semanticweb.org/deliverables/documents/D18.pdf`
[14]`http://www.loa-cnr.it/DOLCE.html`
[15]`http://www.ifomis.org/bfo`

Ontology [Smith et al., 2005] does exist), whereas DOLCE-Lite makes heavy use of object properties. More aspects of both foundational ontologies will be addressed in the lecture, and, time permitting, GFO.

To make reuse easier, 'clever modules' of foundational ontologies may be useful, such as light/basic and full versions according to the developers' taste, a separate major branch of the ontology (e.g., using only Endurants), and a computationally better behaved fragment with the best semantic approximation of the full version (i.e., not merely dropping the violating axioms), such as an OWL 2 EL compliant fragment of DOLCE.

Once the foundational ontology is imported into yours, the task is to find the right classes for your domain classes. This is illustrated in the following example, starting with a very basic African Wildlife Ontology, and gradually extending it and improving its quality.

Example 6.2. The African Wildlife Ontology (AWO) is a very basic tutorial ontology of the "A Semantic Web Primer" book by Grigoris Antoniou and Frank van Harmelen [Antoniou and van Harmelen, 2003]. An OWL version of it, `AfricanWildlife-Ontology0.owl`, has 10 classes and 3 object properties concerning animals such as Lion, Giraffe, Plant, and object properties eats and is-part-of, and has plenty of annotations that give an idea of what should be modelled (else: see 4.3.1 pages 119-133 in [Antoniou and van Harmelen, 2003]). Upon running the reasoner to classify the classes and individuals, it will classify, among others, that Carnivore is a subclass of Animal (i.e, $AWO \models \mathsf{Carnivore} \sqsubseteq \mathsf{Animal}$).

This is not really exciting, and the tutorial ontology is not of a particularly good quality. First, we *add* knowledge: proper parthood, a few more plant parts and animals, such as Impala, Warthog, and RockDassie, and *refine* it, such that giraffes eat not only leaves but also twigs. This version of the African Wildlife Ontology is named `AfricanWildlifeOntology1.owl`, and accessible at `http://www.meteck.org/teaching/ontologies/`. With this additional knowledge, warthogs are classified as omnivores, lions as carnivores, giraffes as herbivores, and so on. We still miss out on having impalas classified as herbivores; what can—or should—you add to the ontology to achieve that? That is, what properties, and how, have to be added so that it will deduce that all impalas are herbivores? (not simply by asserting $\mathsf{Impala} \sqsubseteq \mathsf{Herbivore}$, but by using properties of impalas)

Adding classes and object properties to an ontology does not necessarily make a *better quality* ontology. One aspect that does with respect to the subject domain, is to *refine the represented knowledge* and with *more constraints* so as to limit the possible models, such as that giraffes eat both leaves and twigs, and are disjoint from impalas, and adding more characteristics to the object properties, e.g., that the is-part-of is not only transitive, but also reflexive, and is-proper-part-of is transitive and irreflexive or asymmetric (the latter we can add thanks to the increased expressiveness of OWL 2 DL compared to OWL-DL, but not both irreflexivity and asymmetry) (see also Section 6.2).

Another aspect is purely engineering practice: if the intention is to put the ontology online, it should be *named* properly, i.e., the URI has to be set so that its contents can be identified appropriately on the Semantic Web; that is, do not simply use the default URI generated by the tool (e.g., `http://www.semanticweb.org/ontologies/2013/0/Ontology1357204526617.owl`), but specify an appropriate one where the ontology will be published, like `http://www.loa-cnr.it/ontologies/DOLCE-Lite.owl`. Third, we can improve the ontology's quality by using a *foundational ontology*, as mentioned in Section 6.1.2.

A foundational ontology contains basic categories such as IndependentContinuant/

Endurant (roughly: to represent objects) and Occurent/Perdurant (informally: processes), and Quality for representing attributes, and then their respective sub-categories, such as AmountOfMatter, Feature, PhysicalObject, Achievement, Function, and SpatialRegion; see, e.g., DOLCE, BFO, GFO, GUM, and SUMO that were introduced in Section 6.1.1.

For the sake of example, let us take DOLCE to enrich the African Wildlife Ontology. To do this, we need to import into our wildlife ontology an OWLized version of DOLCE; in this case, we import `DOLCE-lite.owl`[16]. Then, consider first the taxonomic component of DOLCE in Figure 6.1-B (for details, see Wonderweb deliverable D18 Fig 2 p14 and Table 1 p15 or explore the imported ontology with its annotations). Where does Plant fit in in the DOLCE categorisation? Giraffes drink water: where should we put Water? Impalas run (fast); where should we put Running? Lions eat impalas, and in the process, the impalas die; where should we put Death? To answer such questions, we have to look at the principal distinctions made in DOLCE among its categories. Let us take Plant: is Plant wholly presents during its existence (enduring), or is it happening in time (perduring)? With a 3D versus 4D worldview, the former applies. Within endurants, we look at its subclasses, which are Arbitrary Sum, Physical Endurant, and Non-Physical Endurant: a plant is certainly not some arbitrary collection of things, like the set of this lecture notes and your pencil are, and a plant takes up physical space, so one chooses Physical Endurant. We repeat this for the subclasses of Physical Endurant, which are Feature, Amount of Matter, and Physical Object. A feature (in DOLCE) is something like a bump in the road or the hole in a swiss cheese, hence quite distinct from Plant (but a plant can have such things). Amount of matter is in natural language normally denoted with a mass noun, such as gold and water, and it can be counted only in quantities (a liter of water); however, plants can be counted, so they are physical objects and, hence, we can add AWO:Plant $\sqsubseteq$ dolce:PhysicalObject to the ontology. One can find the alignments for the other ones in a similar step-wise way. The answers can be found in `AfricanWildlifeOntology2.owl`.

DOLCE is more than a taxonomy, and we can also inspect in more detail its object properties and reuse the properties already defined instead of re-inventing them. First, the African Wildlife Ontology's is-part-of is the same as DOLCE's part-of, and likewise for their respective inverses. Concerning the subject domain, here are a few modelling questions. The Elephant's Tusks (ivory) are made of Apatite (calcium phosphate, an amount of matter); which DOLCE relation can be reused? Giraffes eat leaves and twigs; how do Plant and Twig relate? How would you represent the Size (Height, Weight, etc.) of an average adult elephant; with DOLCE's Quality or an OWL data property? Answers to the former two questions are included in `AfricanWildlifeOntology2.owl`: we have AWO:Tusk $\sqsubseteq$ dolce:PhysicalObject and AWO:Apatite $\sqsubseteq$ dolce:AmountOfMatter, so we need to find an object property that has as domain a physical object and as range an amount of matter; at present, the easiest way to find out, is to run it through the OntoPartS tool [Keet et al., 2012], which returns the constitution relation as the only one that fits these constraints. OntoPartS's constitution is more restrictive than DOLCE's, so we can either 1) use dolce:generic-constituent that relates perdurants or endurants or 2) add AWO:constituted-of with domain and range dolce:PhysicalObject and range dolce:AmountOfMatter and add AWO:constituted-of $\sqsubseteq$ dolce:generic-constituent, and then assert Tusk $\sqsubseteq$ $\exists$constituted-of.Apatite in our ontology. Option 1 has the benefit of direct reuse of a relation from DOLCE instead of inventing our own from scratch, whereas option 2 is more restrictive and precise, thereby also improving the ontology's quality.

How does it work out when we import BFO into `AfricanWildlifeOntology1.owl`?

[16]`http://www.loa-cnr.it/ontologies/DOLCE-Lite.owl`

Aside from minor differences—e.g., Death is not a type of Achievement as in DOLCE, but a ProcessBoundary instead, and animals and plants are subtypes of Object, see also Figure 6.1-A—there is a major difference with respect to the object properties (BFO has none). A possible outcome of linking the wildlife ontology to BFO is included in `AfricanWildlifeOntology3.owl`. To do these last two exercises with DOLCE and BFO in a transparent and reusable way, however, we need a mapping between the two foundational ontologies. Even more so: if there was a proper mapping, probably only one of the two exercises would have sufficed and the software would have taken care of the mappings between the two. Mappings are now available on ROMULUS, and a solid method to 'swap' a foundational ontology is being developed.

One could take the development a step further by adding types of part-whole relations[17] [Keet and Artale, 2008] so as to be more precise than only a generic part-of relation: e.g., Root is a *structural part of* some Plant and NatureReserve is *located-in* some Country, which will be discussed in some detail in the next section. Another option is to consider a Content Ontology Design Pattern[18] [Presutti et al., 2008], such as being more finicky about names for plants and animals with, perhaps, the Linnaean Taxonomy[19] content pattern or adding some information on the Climatic Zone[20] where the plants and animals live, and so on[21]. ODPs are the topic of Section 6.3. ♢

Instead of the tutorial ontology, you may like to inspect a real ontology that is linked to DOLCE. There are multiple examples, one of which is the Data Mining Optimization Ontology we have come across in Chapter 1 [Keet et al., 2015]. You can download the latest ontology from `http://www.dmo-foundry.org` to inspect it in your ODE of choice, and a selection of the links is depicted in Figure 6.4.

Methods and supporting tools are being developed that are informed by foundational ontologies or provide actual support using them, e.g., [Hoehndorf et al., 2010, Khan and Keet, 2012, Keet, 2012b, Keet et al., 2013a, Hepp, 2011], but more can be done to assist the modeller in the ontology authoring process.

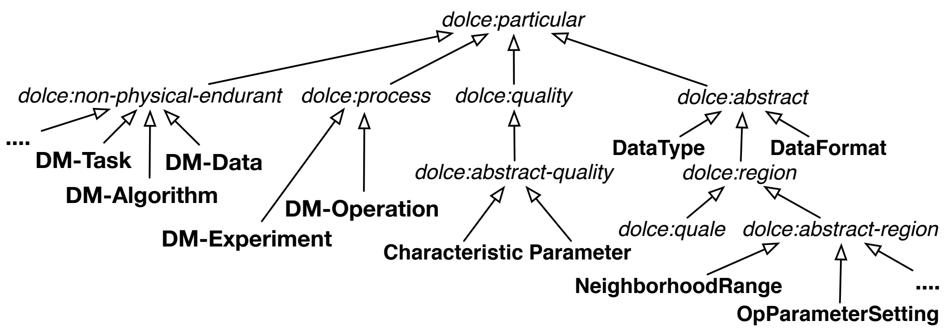


Figure 6.4: Selection of DMOP classes linked to DOLCE.

[17]`http://www.meteck.org/teaching/ontologies/pwrelations.owl`

[18]`http://www.ontologydesignpatterns.org/`

[19]`http://ontologydesignpatterns.org/wiki/Submissions:LinnaeanTaxonomy`

[20]`http://ontologydesignpatterns.org/wiki/Submissions:ClimaticZone`

[21]But note that regarding content, one also can take a bottom-up approach to ontology development with resources such as the Environment Ontology (`http://www.environmentontology.org/`) or pick and choose from 'semantified' Biodiversity Information Standards (`http://www.tdwg.org/`) etc. Bottom-up approaches are the topic of the next chapter.

6.2 Part-whole relations

A, if not *the*, essential relation is the part-whole relation, which is deemed essential by the most active adopters of ontologies—i.e., bio- and medical scientists—while its full potential is yet to be discovered by, among others, manufacturing to manage components of devices. Let us start with some example modelling questions to get an idea of the direction we are heading at:

- Is City a subclass of or a part of Province?
- Is a tunnel part of the mountain? If so, is it a 'part' in the same way as the sand of your sandcastle on the beach?
- What is the difference, if any, between how Cell nucleus and Cell are related and how Receptor and Cell wall are related? Or between the circuit on the ethernet card embedded on the motherboard and the motherboard in the computer?
- Assuming boxers must have their own hands and boxers are humans, is Hand part of Boxer in the same way as Brain is part of Human?
- Consider that "Hand is part of Musician" and "Musician part of Orchestra". Clearly, the musician's hands are not part of the orchestra. Is part-of then not transitive, or is there a problem with the example?

To shed light on part-whole relations in its broadest sense and sort out such modelling problems, we will look first at mereology (the Ontology take on part-whole relations), and to a lesser extent meronymy (from linguistics), and subsequently structure the different terms that are perceived to have something to do with part-whole relations into a taxonomy of part-whole relations, based on [Keet and Artale, 2008].

6.2.1 Mereology

Let us briefly look at the most 'simple' mereological theory, Ground Mereology. We take the one where part-of is primitive (though we also can take proper parthood as primitive and define parthood in terms of it [Varzi, 2004]). Parthood is reflexive (everything is part of itself, Eq. 6.1), antisymmetric (two distinct things cannot be part of each other, or: if they are, then they are the same thing, Eq. 6.2), and transitive (if x is part of y and y is part of z, then x is part of z, Eq. 6.3):

$$\forall x(part\_of(x,x)) \tag{6.1}$$

$$\forall x,y((part\_of(x,y) \land part\_of(y,x)) \to x = y) \tag{6.2}$$

$$\forall x,y,z((part\_of(x,y) \land part\_of(y,z)) \to part\_of(x,z)) \tag{6.3}$$

With these axioms, on can define *proper parthood*:

$$\forall x,y(proper\_part\_of(x,y) \equiv part\_of(x,y) \land \neg part\_of(y,x)) \tag{6.4}$$

Proper parthood is transitive (Eq. 6.5), asymmetric (if x is part of y then y is not part of x, Eq. 6.6) and irreflexive (x is not part of itself, Eq. 6.7):

$$\forall x,y,z((proper\_part\_of(x,y) \land proper\_part\_of(y,z)) \to proper\_part\_of(x,z)) \tag{6.5}$$

$$\forall x,y(proper\_part\_of(x,y) \to \neg proper\_part\_of(y,x)) \tag{6.6}$$

$$\forall x \neg(proper\_part\_of(x,x)) \tag{6.7}$$

These basic axioms already enables us to define several other common relations.
Overlap (x and y share a piece z):

$$\forall x,y(overlap(x,y) \equiv \exists z(part\_of(z,x) \land part\_of(z,y))) \tag{6.8}$$

Underlap (x and y are both part of some z):

$$\forall x, y(underlap(x, y) \equiv \exists z(part\_of(x, z) \wedge part\_of(y, z))) \tag{6.9}$$

Over- & undercross (over/underlap but not part of):

$$\forall x, y(overcross(x, y) \equiv overlap(x, y) \wedge \neg part\_of(x, y)) \tag{6.10}$$

$$\forall x, y(undercross(x, y) \equiv underlap(x, y) \wedge \neg part\_of(y, x)) \tag{6.11}$$

Proper overlap & Proper underlap:

$$\forall x, y(p\_overlap(x, y) \equiv overcross(x, y) \wedge overcross(y, x)) \tag{6.12}$$

$$\forall x, y(p\_underlap(x, y) \equiv undercross(x, y) \wedge undercross(y, x)) \tag{6.13}$$

But there are 'gaps', some would say. Among others: with x as part, what to do with the 'remainder' that makes up y? There are two options:

- Weak supplementation: every proper part must be supplemented by another, disjoint, part, resulting in Minimal Mereology (MM).
- Strong supplementation: if an object fails to include another among its parts, then there must be a remainder, resulting in Extensional Mereology (EM).

There is a problem with EM, however: non-atomic objects with the same proper parts are identical (extensionality principle), but sameness of parts may not be sufficient for identity. For instance, two objects can be distinct purely based on arrangement of its parts, like there is a difference between statue and its marble, several flowers and a bouquet of flowers. This is addressed in General Extensional Mereology (GEM); see also Figure 6.5.

One can wonder if parthood goes on infinitely, or if there is instead some 'basic element', called Atom, and some ultimate whole that encompasses everything. These different commitments generate additional mereological theories. Moreover, we could *temporalise* each mereological theory, and from a modelling perspective in the context of ontologies as engineering artefacts, they are still inadequate, as we shall see in the next section.

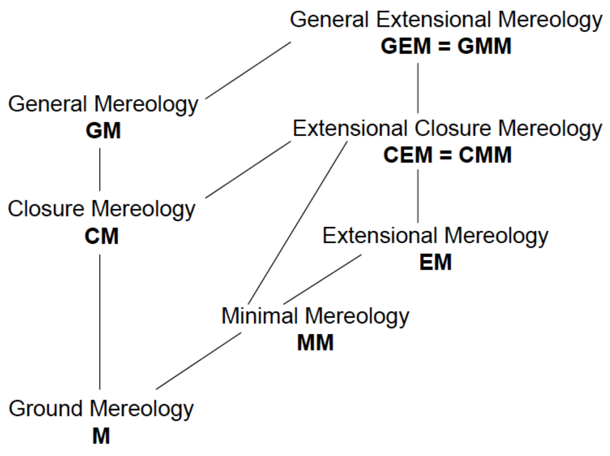


Figure 6.5: Hasse diagram of mereological theories; from weaker to stronger, going uphill (after [Varzi, 2004]). Atomicity can be added to each one.

6.2.2 Modelling and reasoning in the context of ontologies

Mereology is, however, not enough for ontology engineering. Partially due to the 'spill-over' from conceptual data modelling and cognitive science, a whole range of relations are sometimes referred to as a parthood relation, but which are not upon closer inspection. This has been investigated by several researchers. We shall take a closer look at Keet and Artale's taxonomy of part-whole relations [Keet and Artale, 2008] during the lecture; the informal graphical rendering is depicted in Figure 6.6.

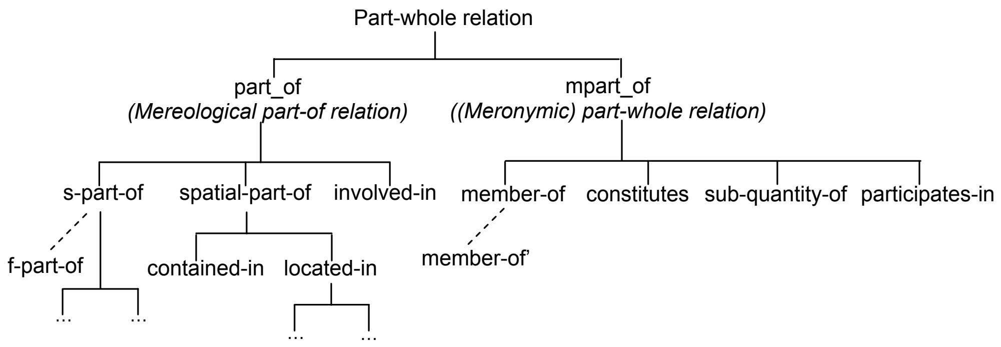


Figure 6.6: Taxonomy of basic mereological (left-hand branch) and meronymic (right-hand branch) part-whole relations; s-part-of = structural part-of; dashed lines: the subtype has additional constraints on the participation of the entity types; ellipses: possible finer-grained extensions to the basic part-whole relations. (Source: [Keet and Artale, 2008])

This, in turn, can be put to use with manual or software-supported guidelines, such as OntoPartS, to choose the most appropriate part-whole relation for the modelling problem at hand[22]. However, the fancy mereological theories from philosophy are, as of yet, practically not feasible to implement, and there is no DL or OWL that actually allows one to represent all of even the most basic mereological theory (Ground Mereology). We shall take a look at some of the representation and reasoning trade-offs with respect to the OWL 2 species in the lecture.

Table 6.2: Properties of parthood and proper parthood compared to their support in $\mathcal{DLR}_\mu$, $\mathcal{SHOIN}$ and $\mathcal{SROIQ}$. *: properties of the parthood relation (in M); ‡: properties of the proper parthood relation (in M).

Language ⇒ **Feature** ⇓	$\mathcal{DLR}_\mu$	$\mathcal{SHOIN}$ (∼ OWL-DL)	$\mathcal{SROIQ}$ (∼ OWL 2 DL)	DL-Lite$_A$ (∼ OWL 2 QL)
Reflexivity *	+	−	+	−
Antisymmetry *	−	−	−	−
Transitivity * ‡	+	+	+	−
Asymmetry ‡	+	+	+	+
Irreflexivity ‡	+	−	+	−
Acyclicity	+	−	−	−

The representation issues affect what we can obtain with automated reasoning over part-whole relations. Let us first consider what can be represented in DLs and OWL species, which is shown in Table 6.2, and for OWL 2 one has to recollect the restrictions on combinations of property characteristics (Section 4.2.1).

[22]`http://www.meteck.org/files/ontopartssup/supindex.html`

Now, what kind of things can be derived with the part-whole relations, and what use may it have? Informally, e.g., when we can deduce which part of the device is broken, then only that part has to be replaced instead of the whole it is part of (saving a company money), and one may want to deduce that when I have an injury in my ankle, I have an injury in my limb (but not deduce that if you have an amputation of your toe, you also have an amputation of your foot that the toe is (well, was) part of). If a toddler swallowed a Lego brick, it is spatially *contained in* his stomach, but one does not deduce it is structurally *part* of his stomach (normally it will leave the body unchanged through the usual channel). This toddler-with-lego-brick gives a clue why, from an ontological perspective, equation 23 in [Cuenca Grau et al., 2008] is incorrect (in addition to being prohibited by OWL 2 DL anyway). Or, e.g., the part-of relation is reflexive, but note that one cannot represent reflexivity in OWL 2 RL. A consequence of asserting reflexivity in the ontology is that then for a domain axiom like each Twig is a part-of some Plant, one deduces that each Twig is a part-of some Twig as well, which is an uninteresting deduction, and, in fact, points to a defect: it should have been asserted to be a *proper* part—which is irreflexive—of Plant. Reflexivity, however, is very useful in other cases, such as the connection relation for spatial relations.

A separate issue that the solution proposed in [Keet and Artale, 2008] brought afore, is that it actually requires one to declare the taxonomy of relations correctly. This can be done by availing of the so-called *RBox Reasoning Service* [Keet and Artale, 2008], which can be considered an *ontological* reasoning service. We shall look at its motivation and workings in the lecture. The basic idea is that the hierarchy of object properties must be well-formed, in that in every model, the tuples of the sub-property are a subset of the tuples of its parent property. The simplest version of this is that the domain and/or range of the sub-property must be a subclass of the domain and/or range of its super-property. More formally, we introduce first some notation to denote the user-defined domain and range of an object property:

Definition 6.1. *(User-defined Domain and Range Concepts [Keet and Artale, 2008]). Let R be a role and $R \sqsubseteq C_1 \times C_2$ its associated Domain & Range axiom. Then, with the symbol D_R we indicate the* User-defined Domain *of R—i.e., $D_R = C_1$—while with the symbol R_R we indicate the* User-defined Range *of R—i.e., $R_R = C_2$.*

The RBox Compatibility can then be defined as follows, covering each permutation of domain and range of the sub-and super property in the hierarchy in the role box $\mathcal{R}$:

Definition 6.2. *(RBox Compatibility [Keet and Artale, 2008]) For each pair of roles, R, S, such that $\langle \mathcal{T}, \mathcal{R} \rangle \models R \sqsubseteq S$, check whether:*

Test 1. $\langle \mathcal{T}, \mathcal{R} \rangle \models D_R \sqsubseteq D_S$ *and* $\langle \mathcal{T}, \mathcal{R} \rangle \models R_R \sqsubseteq R_S$*;*

Test 2. $\langle \mathcal{T}, \mathcal{R} \rangle \not\models D_S \sqsubseteq D_R$*;*

Test 3. $\langle \mathcal{T}, \mathcal{R} \rangle \not\models R_S \sqsubseteq R_R$*.*

An RBox is said to be compatible iff `Test 1` *and (*2 *or* 3*) hold for all pairs of role-subrole in the RBox.*

Note that checking for RBox compatibility—hence, for ontological RBox correctness—can be implemented by availing of the standard DL/OWL automated subsumption reasoning service and an extension to this basic RBox compatibility covering all role inclusion axioms (OWL object property expressions) has been proposed recently with the *SubProS* and *ProChainS* reasoning services for OWL 2 ontologies [Keet, 2012a].

The part-whole taxonomy, the *RBox Compatibility* service, and the OntoPartS[23]

[23]`http://www.meteck.org/files/ontopartssup/supindex.html`

tool's functionalities do not solve all modelling problems of part-whole relations, but at least provide you with a sound basis and some guidelines.

Various extensions to mereology are being investigated, such as mereotopology, the notion of essential parthood [Artale et al., 2008], and mereogeometry. We shall touch upon basic aspects for mereotopology; the interested reader may want to consult, among others, ontological foundations [Varzi, 2007] and its applicability and modelling aspects in the Semantic Web setting with OWL ontologies [Keet et al., 2012], the introduction of the RCC8 spatial relations [Randell et al., 1992], and exploration toward integrating RCC8 with OWL [Grütter and Bauer-Messmer, 2007, Stocker and Sirin, 2009].

6.3 Ontology Design Patterns

A different approach to reuse, is to avail of *ontology design patterns* (ODPs), which is inspired by the idea of software design patterns. The underlying assumption and experience is that it is hard to reuse only the useful pieces of a comprehensive (foundational) ontology, and the cost of reuse may be higher than developing a new ontology from scratch (but recollect [Keet, 2011b]); hence, there is perceived to be the need for small (or cleverly modularized) ontologies with explicit documentation of design rationales, and best reengineering practices, which can be addressed with ODPs. In short, such ODPs summarize the good practices that are to be applied within design solutions and they keep track of the design rationales that have motivated their adoption [Presutti et al., 2008]. Basically, ODPs provide mini-ontologies with formalised knowledge for how to go about modelling reusable pieces, e.g. an n-ary relation or a relation between data type values, in an ontology (in OWL-DL), so that one can do that consistently throughout the ontology development and across ontologies. ODPs for specific subject domains are called content ODPs, such as the 'sales and purchase order contracts' or the 'agent role' to represent agents, the roles they play, and the relations between them. [Presutti et al., 2008]

There are several different types of ODPs, which are summarized in Figure 6.7, which was taken from the rather comprehensive deliverable about ODPs [Presutti et al., 2008]. There are six families of ODPs: Structural OPs, Correspondence OPs, Content OPs (CPs), Reasoning OPs, Presentation OPs, and Lexico-Syntactic OPs. We shall look into the ODP briefly during the lecture.

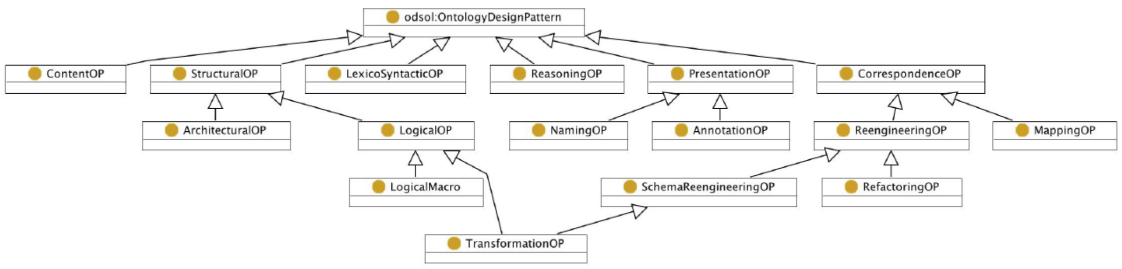


Figure 6.7: Categorisation of types of ontology design patterns. (Source: [Presutti et al., 2008]).

Other foundational ontology aspects, such as philosophy of language, modal logic, change in time, properties, and dependence, will not be addressed in this course. The free online Stanford Encyclopedia of Philosophy[24] contains comprehensive, entry-level readable, overviews of such foundational issues.

[24]`http://plato.stanford.edu/`

6.4 Exercises

Exercise 27. What are the major differences between DOLCE and BFO in terms of philosophical approach?

Exercise 28. What is the major difference between DOLCE and BFO in type of contents of the ontologies?

Exercise 29. Content comparison:

a. Try to match the DOLCE classes Endurant, Process, Quality, Amount of Matter, Accomplishment, Spatial Region, Agentive Physical Object, and Set to a class in BFO.
b. If you cannot find a (near) equivalence, perhaps as a subclass-of some BFO class? And if not even that, why do you think that (those) class(es) is (are) not mappable?

Exercise 30. Give the pros and cons of having a separate relation ontology.

Exercise 31. Find at least 2 common relations—in terms of definition or description and intention—in the OWLized DOLCE, GFO and RO.

Exercise 32. Assume you are asked to develop an ontology about

a. Sociological and organizational aspects of public administration
b. The physiology and chemistry of medicinal plants
c. A topic of your choice

Which (if any) foundational ontology would you choose for each one? Why?

Exercise 33. Download ONSET from `http://www.meteck.org/files/onset/` and redo Exercise 32, but now use the ONSET tool to obtain an answer. Does it make any difference? Were you reasons for choosing a foundational ontology the same as ONSET's?

Exercise 34. Consider the following scenario.

> Both before and since the 2008 recession hit, banks have been merging and buying up other banks, which have yet to integrate their IT systems within each of the consolidated banks, and meet new regulations on transparency of business operations. To achieve that, you are tasked with developing an ontology of banks that will facilitate the database integration and transparency requirements. In such an ontology there will be concrete entities e.g., Bank manager and ATM, and abstract entities e.g., Loans. For this to be possible, the ontological assumptions that are made by the ontology must be based on human common-sense. Processes, such as withdrawals and deposits must also be modelled. It must be possible to capture dates and times for operations that occur between entities and processes. Past and present transactions must be allowed in the ontology. Entities of the ontology may have properties and values associated with them e.g., an individual has a credit rating. You are required to provide your lecturer with the axioms found in your ontology therefore it will be useful refer to or possibly use components of an ontology that implements a particular mereology theory such as classical extensional mereology (CEM) or any other. This ontology must be represented in OWL 2 DL.

Which (if any) foundational ontology would you choose? Why?

Exercise 35. Download either AfricanWildlifeOntology2.owl (with DOLCE) or African-WildlifeOntology3.owl (with BFO) from `http://www.meteck.org/teaching/ontologies/`, open it in the ontology development environment of choice, and inspect its contents. Modify the African Wildlife Ontology as follows:

a. Add enough knowledge so that RockDassie will be classified automatically as a subclass of Herbivore.
b. Add information that captures that lions, impalas, and monkeys live in nature reserves, and that monkeys can also be found on some university campuses.

Exercise 36. There is some ontology O that contains the OWLized DOLCE taxonomy and the following expressions:

$$\begin{array}{ll} \mathsf{hasPart} \sqsubseteq \mathsf{PD} \times \mathsf{PD}, & \mathsf{Human} \sqsubseteq \mathsf{ED}, \mathsf{Brain} \sqsubseteq \mathsf{ED}, \\ \mathsf{involves} \sqsubseteq \mathsf{PT} \times \mathsf{PT}, & \mathsf{Metabolizing} \sqsubseteq \mathsf{PD}, \mathsf{Living} \sqsubseteq \mathsf{PD}, \\ \mathsf{involves} \sqsubseteq \mathsf{hasPart}, & \mathsf{Human} \sqsubseteq \exists \mathsf{hasPart}.\mathsf{Brain}, \\ \mathsf{Trans}(\mathsf{hasPart}), & \mathsf{Living} \sqsubseteq \exists \mathsf{involves}.\mathsf{Metabolizing}. \end{array}$$

Is Human consistent? Verify this with the reasoner and explain why.

Exercise 37. Consider the two OWL ontologies summarised in Table D.1.

a. What is (are) the deduction(s) by the automated reasoner?
b. Is there a difference in the deductions between the two? If so, why; if not, why not?
c. How can we fix the defect(s)?

(For the dedicated: further details regarding the reasoning service that helps finding the flaw and suggests how this can be resolved can be found in [Keet, 2012a])

Table 6.3: Sample ontologies with some defect(s).

$\mathcal{O}_1$	**OPEs**	**CEs**	**Inferred**
	$\mathtt{R} \sqsubseteq \mathtt{PED} \times \mathtt{PED}$ $\mathtt{S} \sqsubseteq \mathtt{ED} \times \mathtt{ED}$ $\mathtt{S} \sqsubseteq \mathtt{R}$	OWLized DOLCE, $\mathtt{Ed}_1 \sqsubseteq \mathtt{ED}$, $\mathtt{Ed}_2 \sqsubseteq \mathtt{ED}$, $\mathtt{Ped}_1 \sqsubseteq \mathtt{PED}$, $\mathtt{Ped}_2 \sqsubseteq \mathtt{PED}$, $\mathtt{Ed}_1 \sqsubseteq \exists \mathtt{S}.\mathtt{Ed}_2$, $\mathtt{Ped}_1 \sqsubseteq \exists \mathtt{R}.\mathtt{Ped}_2$	?
$\mathcal{O}_2$	**OPEs**	**CEs**	**Inferred**
	as $\mathcal{O}_1$	as $\mathcal{O}_1$, but with $\mathtt{Ed}_2 \sqsubseteq \mathtt{AS}$ (and $\mathtt{PED} \sqsubseteq \neg\mathtt{AS}$ still holds)	?

Exercise 38. OWL permits only binary object properties, though n-aries can be approximated. Describe how they can be approximated, and how your ODP would look like such that, when given to a fellow student, s/he can repeat the modelling of that n-ary exactly the way you did it and add other n-aries in the same way.

Exercise 39. Inspect the Novel Abilities and Disabilities OntoLogy for ENhancing Accessibility: ADOLENA; Figure 6.8 provides a basic informal overview. Can (any of) this be engineered into an ODP? If so, which type(s), how, what information is needed to document an ODP?

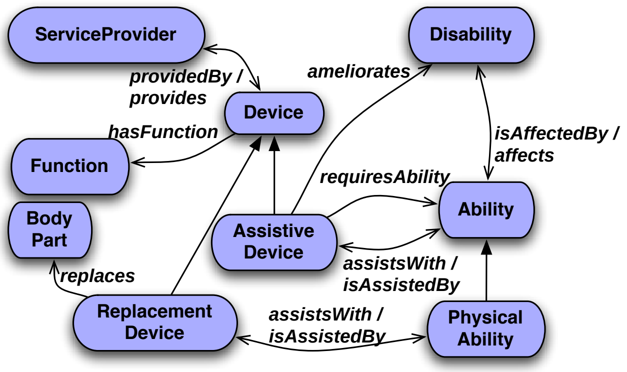


Figure 6.8: Informal view of the ADOLENA ontology.

6.5 Literature and reference material

1. Masolo, C., Borgo, S., Gangemi, A., Guarino, N., Oltramari, A.: WonderWeb Deliverable D18–Ontology library. WonderWeb. `http://wonderweb.semanticweb.org/` (2003). There are also slides of Guarino's OntoLog seminar[25] for a digest of the material.
2. DOLCE in OWL[26] and in various ultralite versions[27], BFO online[28], GFO online[29], The OWL versions of DOLCE and BFO in DL and Manchester syntax in one pdf[30]
3. Keet, C.M. and Artale, A. Representing and Reasoning over a Taxonomy of Part-Whole Relations. *Applied Ontology*, IOS Press, 2008, 3(1-2): 91-110.
4. Presutti, V., Gangemi, A., David, S., de Cea, G. A., Surez-Figueroa, M. C., Montiel-Ponsoda, E., Poveda, M. A library of ontology design patterns: reusable solutions for collaborative design of networked ontologies. NeOn deliverable D2.5.1, Institute of Cognitive Sciences and Technologies (CNR). 2008.
5. An example of combining (linking, integrating, importing) top-level ontologies with domain ontologies: BioTop[31].

[25] `http://ontolog.cim3.net/file/resource/presentation/NicolaGuarino_20060202/DOLCE--NicolaGuarino_20060202.pdf`
[26] `http://www.loa-cnr.it/DOLCE.html`
[27] `http://wiki.loa-cnr.it/index.php/LoaWiki:Ontologies`
[28] `http://www.ifomis.org/bfo`
[29] `http://www.onto-med.de/ontologies/gfo/`
[30] `http://www.meteck.org/files/swt09/DolceliteBFOinDLandMSyntax.pdf`
[31] `http://www.imbi.uni-freiburg.de/ontology/biotop/`

CHAPTER 7

Bottom-up Ontology Development

The other direction of approach developing an ontology without starting with a blank slate, is to reuse exiting data, information or knowledge. A motivation to consider this are the results obtained by Simperl et al [Simperl et al., 2010]: they surveyed 148 ontology development projects, which showed that "domain analysis was shown to have the highest impact on the total effort" of ontology development, "tool support for this activity was very poor", and the "participants shared the view that process guidelines tailored for [specialised domains or in projects relying on end-user contributions] are essential for the success of ontology engineering projects". In other words: the **knowledge acquisition bottleneck**. Methods and tools have been, and are being developed to make it less hard to get the subject domain knowledge out of the experts and into the ontology, e.g., through natural language interfaces and diagrams, but also to make it less taxing on the domain experts by reusing the 'legacy' material they already have to store their information and knowledge. It is the latter we are going to look at in this chapter: **bottom-up ontology development** to get the subject domain knowledge represented in the ontology. We approach it from the other end of the spectrum compared to what we have seen in Chapter 6, being starting from more or less reusable non-ontological sources and develop an ontology from that.

Technologies differ according to their focus for semi-automated bottom-up ontology development:

- Ontology learning to populate the TBox, where the strategies can be subdivided into:
  - transforming information or knowledge represented in one logic language into an OWL species;
  - transforming somewhat structured information into an OWL species;
  - starting at the base.
- Ontology learning to populate the ABox.

The latter is carried out typically by either natural language processing (NLP) or one or more data mining or machine learning techniques. In the remainder of this chapter, however, we shall focus primarily on populating the TBox. Practically, this means taking some 'legacy' material (i.e., not-Semantic Web and, mostly, not-ontology) and convert it into an OWL with some manual pre- and/or post-processing. Input artefacts include, but are not limited to:

1. Databases

2. Conceptual data models (ER, UML)
3. Frame-based systems
4. OBO format ontologies
5. Thesauri
6. Biological models
7. Excel sheets
8. Tagging, folksonomies
9. Output of text mining, machine learning, clustering

It is not equally easy (or difficult) to transform them into a domain ontology. Figure 7.1 gives an idea as to how far one has to 'travel' from the legacy representation to a 'Semantic Web compliant' one. The further the starting point is to the left of the figure, the more effort one has to put into realizing the *ontology learning* such that the result is actually usable without the need of a full redesign. Due to the limited time available in this course, we shall not discuss all variants. We shall focus on using a databases as source material to develop an ontology[1], thesauri, and NLP.

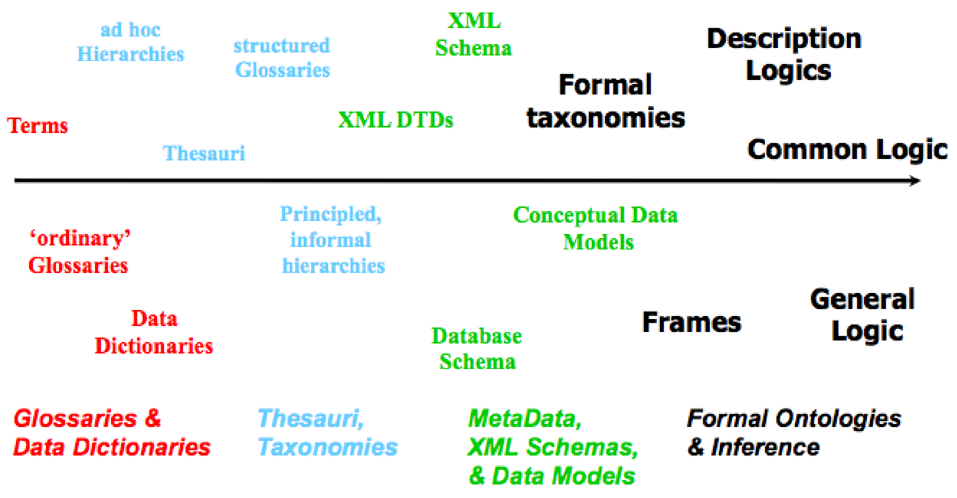


Figure 7.1: Various types of less and more comprehensively formalised 'legacy' resource.

7.1 Relational databases and related 'legacy' KR

The starting position for leveraging the knowledge encoded in a relational database to develop an ontology could be its conceptual data model. However, despite academics' best efforts to teach good design and maintenance methodologies in a degree programme, it is not uncommon in organisations that if there was a conceptual model for the database at all, it is outdated by the time you would want to use it for ontology development. New columns and tables may have been added in the database, constraints removed, tables joined (further denormalised) for better performance or vice versa for cleaner data, and so on, and no-one bothered to go back to the original conceptual, or even relational, model and update it with the changes made. Practically, there likely will be a database with multiple tables that have many (15-50) columns. This is represented at the bottom of Figure 7.2.

[1]Converting a relational database into an RDF triple store is quite common as well, but not covered in the lecture; e.g., the D2R database to RDF mapping (`http://sites.wiwiss.fu-berlin.de/suhl/bizer/d2r-server/`).

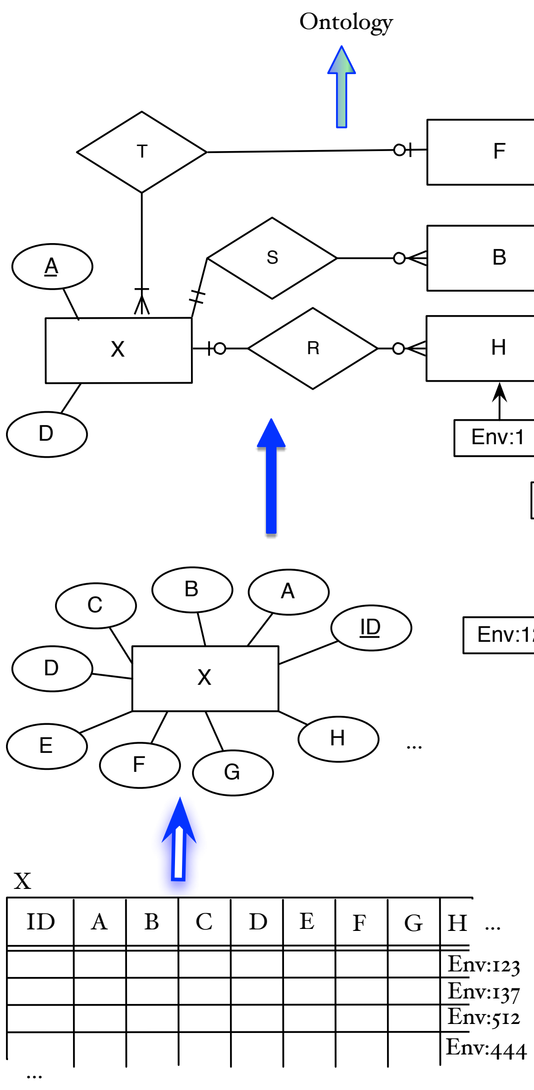


Figure 7.2: Denormalised relational database (bottom), where each table is reverse engineered into an entity in a 'flat' EER diagram (middle), and subsequently refined w.r.t. the hidden entity types and annotations, such as the GO (top), which then finally can be transformed/translated into an ontology.

If we were to simply convert that physical schema into an OWL ontology, the outcome would be a bunch of classes with many data properties and a unnamed object property between a subset of the classes from foreign keys. This won't do as an ontology.

Reverse engineering the database

There are several reverse engineering tools for physical schemas of relational databases, where a first pass results in one of the possible logical models (i.e., the relational model for an RDBMSs), and another iteration brings one up to the conceptual data model (such as ER, ORM) [Hainaut et al., 1993]. This first draft version of the EER model is depicted in EER bubble notation in Figure 7.2, where each table (relation) has become an entity type and each column an attribute. The main problematic consequence for reverse engineering the conceptual data model to feed into an OWL ontology is that the database structure has been 'flattened out', which, if simply reverse engineered, ends up in the 'ontology' as a class with umpteen attributes with which one can do minimal (if at all) automated reasoning (see the first diagram above the table in Figure 7.2).

To avoid this, should one perform some normalization steps to try to get some structure back into the conceptual view of the data alike in the diagram at the top in Figure 7.2, and if so, how? Whether done manually or automatically, it can be cleaned up, and original entity types (re-)introduced, relationships added, and the attributes separated accordingly, thereby making some knowledge implicit in the database schema explicit, which is depicted in the top-half of Figure 7.2. This can be done at least partially automatically by discovering functional dependencies in the data stored in the database tables. Such reverse engineering opens up other opportunities, for one could use such a procedure to also establish some mechanism to keep a 'link' between the terms in the ontology and the source in the database so that the ontology can be used to enhance data analysis through conceptual model or ontology-based querying. A particular algorithm up to obtaining a DL-formalised conceptual data model based on a fully normalised database can be found in, e.g., [Lubyte and Tessaris, 2009].

Figure 7.2 may give the impression that it is easy to do, but it is not. Difficulties achieving it have to do with the formal apparatus of the representation languages[2], and the static linking between the layers and the procedures, which are conveniently depicted with the three arrows, hide the real complexity of the algorithms. Reverse engineering is not simply running the forward algorithm backwards, but has a heuristics component to second-guess what the developers' design decisions may have been along the stages toward implementation and may have a machine learning algorithm to find constraints among instances. Most solutions to date set aside data duplication, violations of integrity constraints, hacks, outdated imports from other databases and assume to have a well-designed relational database in at least 3NF or BCNF, and, thus, the results are imperfect.

In addition to this procedure, one has to analyse the data stored in the database on its exact meaning. In particular, one may come across data in the database—mathematically instances—that are actually assumed to be concepts/universals/classes, whereas others represent real instances. For instance, a Content Management System, such as Joomla, requires the content provider to store a document under a certain category that is considered a class by its user, which, however is stored in a cell of a row in the back-end database, hence, mathematically an instance in the software. Somehow, we need to find that and extract it for use in the ontology in a way that they will become classes. Another typical case is where a structured controlled vocabulary, such as the Gene Ontology we have seen in Section 1.3, has been used in the database for annotation. This is depicted on the right-hand side with `Env:444` and so on. Knowing this, one can reverse engineer that section of the database into a taxonomy in the conceptual data model (shown in Figure 7.2 in the top figure on the right-hand side). Finally, there is a so-called 'impedance mismatch' between database *values* and ABox *objects*, but this is relevant mainly for ontology-based data access (see Chapter 8). So, to convert a database, we end up having some, or all, data where the values are actually concepts that should become OWL classes and values that should become OWL individuals.

[2]For conceptual data modelling languages, among others, the Object Management Group's Ontology definition metamodel (`http://www.omg.org/spec/ODM/1.0/`) is exploring interactions between UML and OWL & RDF, and there are various results on mapping ER, EER, UML, ORM and/or ORM2 into a suitable or convenient DL language [Artale et al., 2007a, Berardi et al., 2005, Calvanese et al., 1998a, Calvanese et al., 1999, Keet, 2008, Keet, 2009b, Keet, 2013].

Enhancing and converting the conceptual model

Having completed all the reverse engineering and data analysis, one can commence with the ontological analysis. Whilst improving the conceptual data model, one could perhaps add a section of another ontology for use, improve on the naming and meaning of the relationships (perhaps one is intended the same as one from a foundational ontology), add constraints, and so forth, which is a separate line of work (see Section 9.2). Subsequently, it will have to be converted to a suitable logic language (in the current setting: an OWL species). One also could switch these steps by first converting it to OWL and then perform the ontological analysis. There are several tools that convert a conceptual model, especially UML Class Diagrams, into OWL, but they have only partial coverage and its algorithms are unclear; for instance, on how one should transform ternaries and what to do with the attributes (recall Section 6.1.1). In addition, they work only with a subset of UML diagrams due to the differences in UML tool implementations (which is due to ambiguity emanating from the OMG standard and differences across versions).

Other languages and OWL

Imperfect transformations from other languages, such as the common OBO format [Golbreich and Horrocks, 2007, Hoehndorf et al., 2010] and a pure frames-based approach [Zhang et al., 2006], are available, which also describe the challenges to create them.

OBO is a Directed Acyclic Graph mainly for classes and a few relationships (mainly is_a and part_of), which relatively easily can be mapped into OWL, and the extras (a.o., date, saved by, remark) could go in OWL's annotations. There are a few mismatches and 'work-arounds', such as the not-necessary and inverse-necessary, and a non-mappable antisymmetry (cannot be represented in OWL). As a result, there are several OBO-in-OWL mappings, some being more comprehensive than others. The latest/official mapping available from `http://oboformat.org` (superseding the earlier mapping by [Golbreich and Horrocks, 2007]), which is also implemented in the OWL API. Most OBO ontologies now also have an OWL version (consult OBO Foundry, BioPortal), but keep both, for each has their advantages (at present). There is one salient difference between OWL and OBO ontologies—more precisely: the approach to modelling—which also affects multilingual ontologies (Section 7.3.3), and how you can view an OBO ontology in Protégé. In OWL, you typically give a class a human readable *name*, whereas in OBO, a class is assigned an *identifier* and the name is associated to that with a label (OBO people who moved to OWL maintain that practice, so numbers as class names do not imply it was natively an OBO ontology). Newer versions of ontology editors let you choose how to render the ontology in the interface, by name or by label. If you find an ontology online and the class names are something alike `IAO12345`, then it was likely an OBO ontology converted into OWL, and you'll have to change the view in the 'ontology view' settings, so that it will show the labels instead of those meaningless numbers.

While OBO and the older frames-based Protégé do serve a user base, their overall impact on widespread bottom-up ontology development for the Semantic Web is very likely to be less than the potential that might possibly be unlocked with leveraging knowledge of existing (relational) databases. One may be led to assume this holds, too, for text processing (NLP) as starting point for semi-automated ontology development, but the results have not been encouraging yet, for various reasons.

7.2 Thesauri

Two bottom-up approaches that, by basic idea at least, can have a large impact on domain ontology development are: taking as basis biological models or any other structured graphical representation with subject domain icons [Keet, 2005, Keet, 2012b] and the rather more cumbersome one of reusing thesauri [Biasiotti and Fernández-Barrera, 2009, Soergel et al., 2004]. Both examples have abundant similar instances in biology, medicine, industry, education, agriculture, cultural heritage, and, undoubtedly, some more automation to realise it would be a welcome addition to ease the efforts to realise the Semantic Web.

There are many thesauri that all revolve around the core notions of **BT** broader term, **NT** narrower term, and **RT** related term (and auxiliary ones UF/USE). For instance, the Educational Resources Information Center thesaurus:

```
reading ability
  BT ability
  RT reading
  RT perception
```

and the AGROVOC of the FAO:

```
milk
  NT cow milk
  NT milk fat
```

How to go from this to an ontology? Three approaches exists (thus far):

- Automatically translate the 'legacy' representation of the ontology into an owl file and call it an ontology (by virtue of being represented in OWL, regardless the content);
- Find some conversion rules that are informed by the subject domain and foundational ontologies (e.g., introducing part-of, made-from, etc.);
- Give up on the idea of converting it into an ontology and settle for the W3C-standardised Simple Knowledge Organisation System[3] format to achieve compatibility with other Semantic Web Technologies.

We will look at the problems with the first option, and achievements with the second and third option.

Problems

The main problems to address are that thesauri are generally a lexicalisation of a conceptualisation and they have low ontological precision with respect to the categories and the relations (see [Soergel et al., 2004] for further details). Mainly, they lack basic categories alike those in DOLCE and BFO (ED, PD, SDC, etc.), the 'RT' can be anything from parthood to transformation, to participation, and anything else, and BT/NT is not the same as *is_a*; hence, the relations are overloaded with (ambiguous) subject domain semantics, and resulting from that, those relationships are used inconsistently—or at least not precise enough for an ontology. For instance, in the aforementioned example, `milk` and `milk fat` relate in a different way to each other than `milk` and `cow milk`, for `milk fat` is a component of milk and `cow milk` indicates its origin (and, arguably, it is part of the cow). Among other approaches, [Soergel et al., 2004] use a 'rules as you go' approach, which we shall look at, but in any case it requires substantial manual intervention.

[3] `http://www.w3.org/2004/02/skos/`

SKOS

Thesauri tend to be very large, and it may well be too much effort to convert them into a real ontology, yet one still would want to have some interoperation of thesauri with other systems so as to avail of the large amounts of information they contain. To this end, the W3C developed a standard called *Simple Knowledge Organisation System(s): SKOS*[4] [Miles and Bechhofer, 2009]. More broadly, it is intended for converting thesauri, classification schemes, taxonomies, subject headings etc. into one interoperable syntax, thereby enabling concept-based search instead of text-based search, reuse of each other's concept definitions, facilitate the ability to search across institution boundaries, and to use standard software. This is a step forward compared to the isolated thesauri.

However, there are also some limitations to it: 'unusual' concept schemes do not fit into SKOS because sometimes the original structure too complex, `skos:Concept` is without clear properties like in OWL, there is still much subject domain semantics in the natural language text which makes it less amenable to advanced computer processing, and the SKOS 'semantic relations' have little semantics, as `skos:narrower` does not guarantee it is *is_a* or *part_of* (it just is the standardised version of NT).

Then there is a peculiarity in the encoding. Let us take the example where Enzyme is a subtype of Protein, hence, we declare:

```
SKOSPaths:protein rdf:type skos:Concept
SKOSPaths:enzyme rdf:type skos:Concept
SKOSPaths:enzyme SKOSPaths:broaderGeneric SKOSPaths:protein
```

in the `SKOSPaths` SKOS file, which are, mathematically, statements about instances. This holds true also if we were to transform an OWL file to SKOS: each OWL class becomes a SKOS instance due to the mapping of `skos:Concept` to `owl:Class` [Isaac and Summers, 2009]. This is a design decision of SKOS. From a purely technical point of view, that can be dealt with easily, but one has to be aware of it when developing applications.

We will look into SKOS during the lecture.

7.3 Natural language and ontologies

There is a lot to be said about how Ontology, ontologies, and natural language interact from a philosophical perspective up to the point that different commitments lead to different features and, moreover, limitations of a (Semantic Web) application; also here more emphasis will be put on an engineering perspective. This section groups together various aspects of this combination, where subsection 7.3.1 is about partially automated bottom-up population and Section 7.3.2 manual construction and validation with domain experts. The last section (Section 7.3.3) is not really about bottom-up, but still has to do with natural languages and ontologies, being multilingualism. (At some point, this section may get its own chapter in the lecture notes.)

7.3.1 NLP for ontology learning

Natural Language Processing (NLP) can be useful for ontology development, be used as a component in an ontology-driven information system, and an NLP application can be enhanced with an ontology. Which approaches and tools suit best depends on the goal (and background) of its developers and prospective users, ontological commitment, and available resources.

[4] `http://www.w3.org/TR/swbp-skos-core-spec`

There are several possibilities for 'something natural language text' and ontologies or ontology-like artifacts:

- Use ontologies to improve NLP: to enhance precision and recall of queries (including enhancing dialogue systems [Vila and Ferrández, 2009]), to sort results of an information retrieval query to the digital library (e.g. GoPubMed[5] [Dietze et al., 2008]), or to navigate online articles (which amounts to linked data[6]).
- Use NLP to develop ontologies (TBox): to search for candidate terms and relations, which is part of the suite of techniques called 'ontology learning', [Coulet et al., 2010, Alexopoulou et al., 2008]; see, e.g. the recent review on NLP and (bio-)ontologies by [Liu et al., 2011].
- Use NLP to populate ontologies with instances (ABox): e.g., document retrieval enhanced by lexicalised ontologies and biomedical text mining [Witte et al., 2007].
- Use ontologies with controlled natural languages in natural language generation (NLG) systems: this can be done using a template-based approach that works quite well for English but much less so for grammatically more structured languages such as Italian [Jarrar et al., 2006] and isiZulu [Keet and Khumalo, 2014b], or with a full-fledged grammar engine as with the Attempto Controlled English[7] and bi-directional mappings (see for a discussion [Schwitter et al., 2008]); see [Bouayad-Agha et al., 2014] for a recent overview.

Intuitively, one may be led to think that simply taking the generic NLP or NLG tools will do fine also for specialised domains, such as (bio-)medicine. Any application does indeed use those techniques and tools[8], but, generally, they do not suffice to obtain 'acceptable' results. Domain specific peculiarities are many and wide-ranging. For instance, to deal with the variations of terms (scientific name, variant, common misspellings) and the grounding step (linking a term to an entity in a biological database) in the ontology-NLP preparation and instance classification [Witte et al., 2007], to characterise the question in a question answering system correctly (e.g., [Vila and Ferrández, 2009]), and to find ways to deal with the rather long strings that denote a biological entity or concept or universal [Alexopoulou et al., 2008].

An example of a comprehensive approach is NLP for pharmacogenomics ontology development, which requires rule construction, 'normalization' of verbs that are candidate relationships, and linking it to the PHARE domain ontology[9] [Coulet et al., 2010]. Some of the mentioned peculiarities actually generate better overall results than in generic or other domain-specific usages of NLP tools, but it requires extra manual preparatory work and a basic understanding of the subject domain and its applications to include also such rules. For instance, enzyme names always end with "-ase", so one can devise a rule with a regular expression to detect these terms ending in "-ase" and add them in the taxonomy as a subclass of Enzyme.

[5]`http://www.gopubmed.com/`

[6]Allen H. Renear and Carole L. Palmer. Strategic Reading, Ontologies, and the Future of Scientific Publishing. Science 325 (5942), 828. [DOI: 10.1126/science.1157784] (but see also some comments on the paper at `http://keet.wordpress.com/2009/09/19/linked-data-as-the-future-of-scientific-publishing/`)

[7]`http://attempto.ifi.uzh.ch/site/`

[8]Paul Buitelaar's slides have examples and many references to NLP tools (`http://www.deri.ie/fileadmin/documents/teaching/tutorials/DERI-Tutorial-NLP.final.pdf`)

[9]`http://bioportal.bioontology.org/ontologies/45138`

7.3.2 Controlled natural languages to interact with domain experts

Controlled natural languages and the broader Natural Language Generation area (NLG) also have both a linguistic component and a computing component. Aside from its use in machine translations, NLG is use to create nice 'syntactic sugar' to the logic-based ontologies, which

- helps the domain experts understand what exactly has been represented in the ontology, and thereby a means to validate that what has been represented is what was intended;
- can be used to write axioms in pseudo-natural language, rather than the logic itself, to develop the ontology.

Thus, it is part of a very bottom-up approach of manual ontology authoring.

Let's start with a few example *verbalisations* in both English and isiZulu:

(S1) Giraffe $\sqsubseteq$ Animal
Each Giraffe is an Animal
izindlulamithi yizilwane

(S2) Herb $\sqsubseteq$ Plant
Each Herb is a Plant
ihebhu ngumuthi

(S3) Milk $\sqcap$ Butter
Milk and Butter
ubisi nebhotela

The underlined text forms part of the pattern, and the vocabulary is inserted on the fly read from the ontology file. For English, it is very well feasible to do this with templates where the terms are slotted in, which can be refined with rules to cate for things such as a vs an. A notable advanced system that does this is Attempto Controlled English[10] [Fuchs et al., 2010]. For many other languages, however, also a comprehensive encoding of a subset of the grammar rules is necessary to generate understandable sentences [Jarrar et al., 2006, Keet and Khumalo, 2014a], such as that $na + i = ne$ for the enumerative 'and' in isiZulu to generate the nebhotela from ibhotela, and much more complex ones for verbs and quantifiers that depend on the noun class of the noun that us used to name the OWL class.

Essentially, each symbol in the language has one or more ways of putting it in natural language, and based on that, one can create a *controlled natural language* that can generate only those sentences following a specified pattern, not some arbitrary sentence that NLP has to deal with. In this way, then a whole ontology can be *verbalised* and presented as (pseudo-)natural language sentences that hide the logic. having that, the next step is to go in the other direction: using the controlled natural language to generate the axioms.

Of course, it takes for granted one already has an ontology in one's preferred language.

7.3.3 Multilingual ontologies

Perhaps this is not the best place to discuss an ontology engineering issue and natural language, but it is easier to grasp now than earlier, as we have come across OWL, OBO, and NLP-driven TBox population now. OWL and its ontology development environment interfaces asks one to use natural language(-like) naming of the classes and object properties. But what to do in a collaborative environment, or integrating or just linking ontologies, and you spelled it hasColour and another developer wrote it as Color in his

[10]http://attempto.ifi.uzh.ch/site/

ontology, or one insists on a euphemism BurglarGuards and another domain expert calls those things PrisonBars? Or you want to develop an ontology that is accessible in all 11 official languages of South Africa?

On the face of it, this may look problematic for OWL, despite that one of its goals was internationalisation. The spelling variants can be handled through annotations and synonyms where the original terms have to be preserved by declaring two classes equivalent. It becomes interesting when one ontology has to serve people who prefer more than one natural language, or at least a natural language different from the ontology. Does one really have to elect one 'core' natural language in which to develop the ontology and sort out 'the rest' with labelling within the ontology, or, worse from a maintenance point of view, create a separate version of the ontology in each language? By now, this has become a 'hot' research topic, and there are several approaches to deal with multilingualism in ontologies. The remainder of the section is more intended to sensitise you to the issues you will have to take into consideration when developing multilingual ontologies, but the state of the art is not yet at the stage where there's one straightforward solution.

The simplest approach is called *semantic tagging*, which is depicted in Figure 7.3: the ontology is 'developed in English' and for other languages, labels are added, such as Fakultät for School. Another approach is to design the ontology in a fundamentally 'language-independent' way. This approach underlies the OBO ontologies: each entity in an OBO-formalised ontology is named with its ID, and then any amount of natural language terms are associated with it.

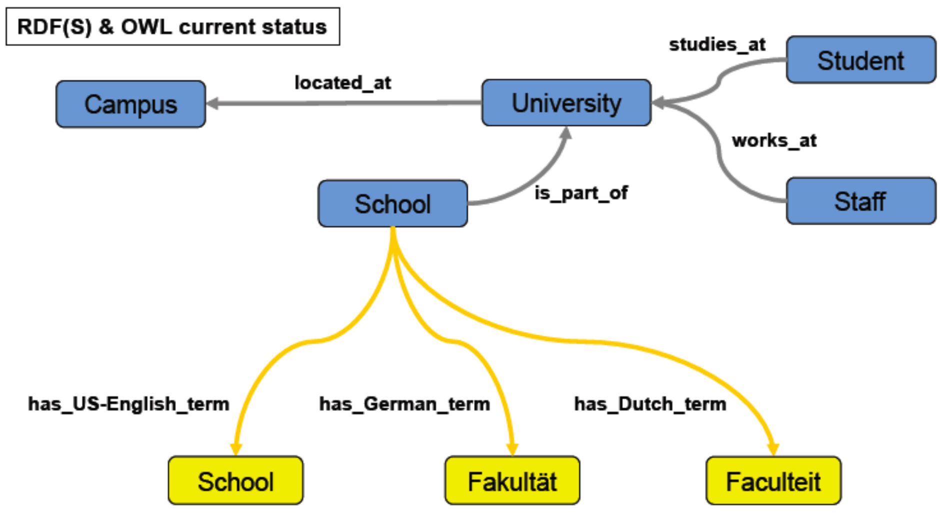


Figure 7.3: Ontologies in practice: Semantic Tagging—Classes, Terms. (Source: http://www.deri.ie/fileadmin/documents/teaching/tutorials/DERI-Tutorial-NLP.final.pdf)

However, both falter as soon as there is no neat 1:1 translation of a term into another single term in a different language—which is quite often the case except for very similar languages—though within the scientific realm, this is much less of an issue, where handling synonyms may be more relevant. The following example illustrates some 'struggling' on how to handle this for when there is not even a name for the entity in the other language (taken from [Alberts et al., 2012])

Example 7.1. Let us consider *ingcula*, which is a "small bladed hunting spear" in isiZulu, but has no equivalent term in English. Trying to represent it in the 'English understanding', then we could have a class Spear that has two properties, e.g., Spear $\sqsubseteq$ $\exists$hasShape.Bladed $\sqcap$ $\exists$participatesIn.Hunting, and then a fuzzy concept to repre-

sent small, following [Bobillo and Straccia, 2011], e.g.,
$\mathsf{MesoscopicSmall} : \mathsf{Natural} \rightarrow [0, 1]$ as a fuzzy datatype,
$\mathsf{MesoscopicSmall(x)} = \mathsf{trz(x, 1, 5, 13, 20)}$ with trz the trapezoidal function,
so that a small spear can be defined as
$\mathsf{SmallSpear} \equiv \mathsf{Spear} \sqcap \exists \mathsf{size.MesoscopicSmall}$
Then we create a class in English and declare something alike
$\mathsf{SmallBladedHuntingSpear} \equiv \mathsf{SmallSpear} \sqcap \exists \mathsf{hasShape.Bladed} \sqcap \exists \mathsf{participatesIn.Hunting}$
This is just one of the possibilities of a formalised translation of an English natural language description, not a definition of *ingcula* as it may appear in an ontology about indigenous knowledge of hunting. But let us assume for now we do want to go in this direction, then we require more advanced capabilities than even lexicalised ontologies, which only link dictionaries and grammars to the ontologies (besides, dictionaries are limited in size and soft copy availability for most South African languages).

A possible solution to this impasse is to use ε-connections between natural language-dependent versions of the ontology. For instance, we could connect inqina.owl#Ingcula@zu to a *set* of axioms or to a dummy class (say, hunting.owl#SmallBladedHuntingSpear@en) with the above-mentioned definition for Ingcula. However, also this is easier said than done, and has the problem of having to maintain an ontology for each language. ♢

To conclude from the example: it is not clear how to handle the multilingualism in this way.

Another idea being proposed is that of a so-called "lexicalised ontology" [Buitelaar et al., 2009], of which an example is depicted in Figure 7.4. Although it still conflates the entity and its preferred label (i.e., name), handling other languages is much more extensive and, at least in theory, will be able to cope with multilingual ontologies to a greater extent. This is thanks to its relatively comprehensive information about the lexical aspects in its own linguistic ontology, with the WordForm etc., which is positioned orthogonally to the domain ontology. In Figure 7.4, the English OralMucosa has its equivalent in German as Mundschleimhaut, which is composed here of two sub-words that are nouns themselves, Mund (mouth) and Schleimhaut (mucosa). This idea has been made more precise and comprehensive in the *Lemon* model that is tailored to the Semantic Web setting [McCrae et al., 2012]. It uses RDF, but we abstract from that here and present only a depiction of the model in Figure 7.5 and an example for the class Cat in Figure 7.6. If you want to know how well this approach works for Bantu languages, have a look at [Chavula and Keet, 2014]; to see FOAF[11] localised in Chichewa check `www.meteck.org/files/ontologies/foaf.ttl`.

Yet a step further in 'decoupling' the ontology aspect from the natural language aspect would be to merge this with OBO's ID approach. At the time of writing this section, it is an active field of research and it will be updated in a next version of the lecture notes.

7.4 Other semi-automated approaches

Other (semi-)automated approaches to bottom-up ontology development include machine learning techniques and deploying so-called 'non-standard' DL reasoning services.

A short overview and relevant references of machine learning techniques for ontology development can be found in [d'Amato et al., 201x], who also outline where such *inductive methods* can be used, being: classifying instances, learning new relationships

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

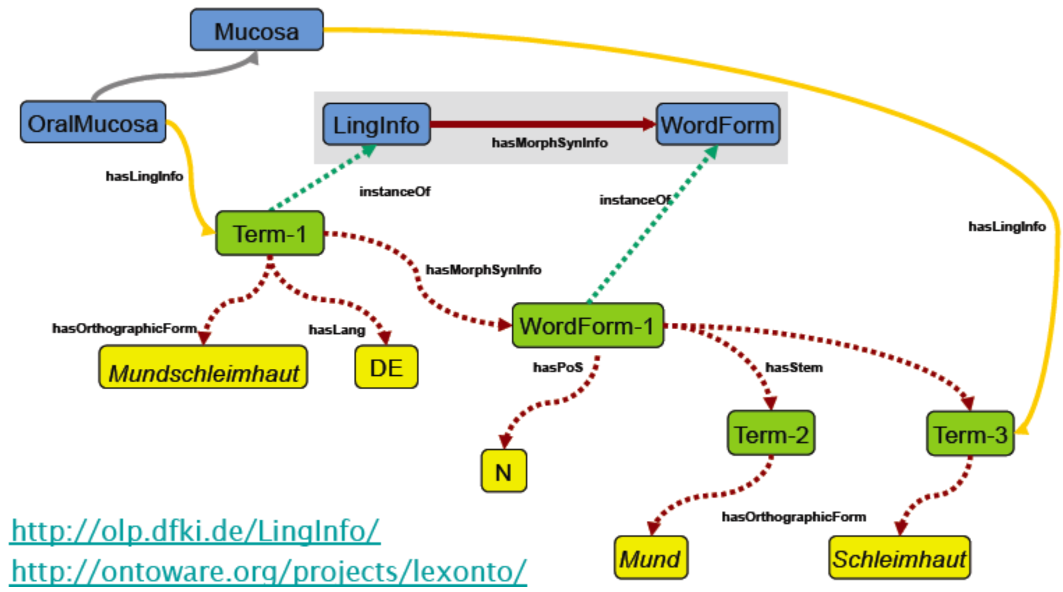


Figure 7.4: Ontologies in practice: Semantic Tagging—Lexicalized Ontologies. (Source: http://www.deri.ie/fileadmin/documents/teaching/tutorials/DERI-Tutorial-NLP.final.pdf)

among individuals, probabilistic ontologies, and probabilistic mapping for the ontology matching task, (semi)-automating the ontology population task, refining ontologies, and reasoning on inconsistent or noisy knowledge bases. Several 'hybrids' exists, such as the linking of Bayesian networks with probabilistic ontologies [da Costa and Laskey, 2006] and improving data mining with an ontology [Zhou et al., 2005].

Other options are to resort to a hybrid of Formal Concept Analysis with OWL [Baader et al., 2007a], least common subsumer [Baader et al., 2007b, Turhan, 2008] [Peñaloza and Turhan, 2011], and similar techniques. The notion of least common subsumer and most specific concept and motivations where and how it may be useful are described in [Peñaloza and Turhan, 2011].

Exploiting biological models

See [Keet, 2005, Keet, 2012b] for semi-automated approaches to formalise the graphical vocabulary in textbooks and drawing tools, and subsequently use an algorithm to populate the TBox with the knowledge taken from the drawings.

See [Hoehndorf et al., 2011] for a first approach in systems biology and simulation, where the authors convert models represented in the Systems Biology Markup Language (SMBL) into OWL.

More TBA.

7.5 Exercises

Exercise 40. Examine Figure 7.7 and answer the following questions.

a. Represent the depicted knowledge in an OWL ontology.
b. Can you represent all knowledge? If not: what not?
c. Are there any problems with the original conceptual data model? If so, which one(s)?

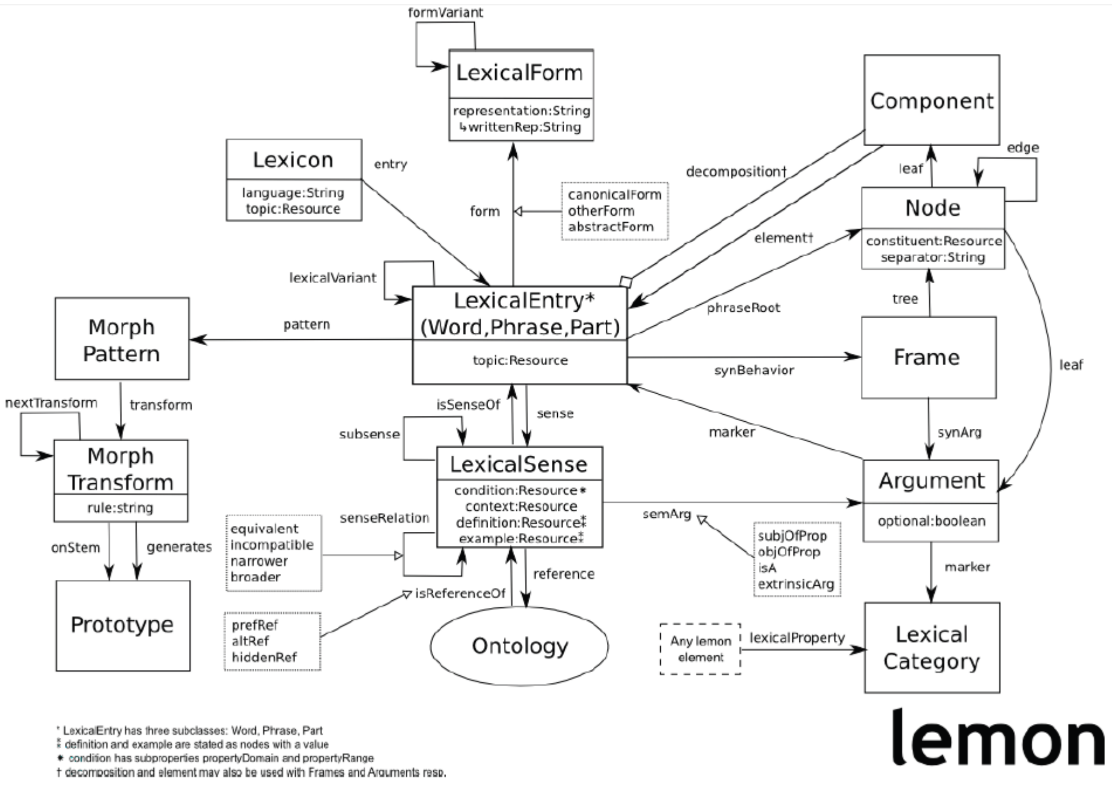


Figure 7.5: The *Lemon* model for multilingual ontologies (Source: [McCrae et al., 2012])

Exercise 41. Figure 7.8 shows a very simple conceptual data model in roughly UML class diagram notation: a partition [read: disjoint, complete] of employees between clerks and managers, plus two more subclasses of employee, namely rich employee and poor employee, that are disjoint from the clerk and the manager classes, respectively (box with cross). All the subclasses have the salary attribute restricted to a string of length 8, except for the clerk entity that has the salary attribute restricted to be a string of length 5. Another conceptual data model, in ORM2 notation (which is a so-called attribute-free language), is depicted in Figure 7.9, which is roughly similar.

a. When you reason over the conceptual data model in Figure 7.8, you will find it has an inconsistent class and one new subsumption relation. Which class is inconsistent and what subsumes what (that is not already explicitly declared)? Try to find out manually, and check your answer by representing the diagram in an OWL ontology and run the reasoner to find out.
b. Develop a proper ontology that can handle both conceptual data models. Consider the issue of how deal with attributes and add the information that clerks work for at most 3 projects and managers manage at least one project.

Exercise 42. Consider the small section of the Educational Resources Information Center thesaurus, below.

a. In which W3C-standardised (Semantic Web) language would you represent it, and why?
b. Are all BT/NT assertions subsumption relations?
c. There is an online tool that provides a semi-automatic approach to developing a domain ontology in OWL starting from SKOS. Find it. Why is it *semi*-automatic and can that be made fully automatic (and if so, how)?

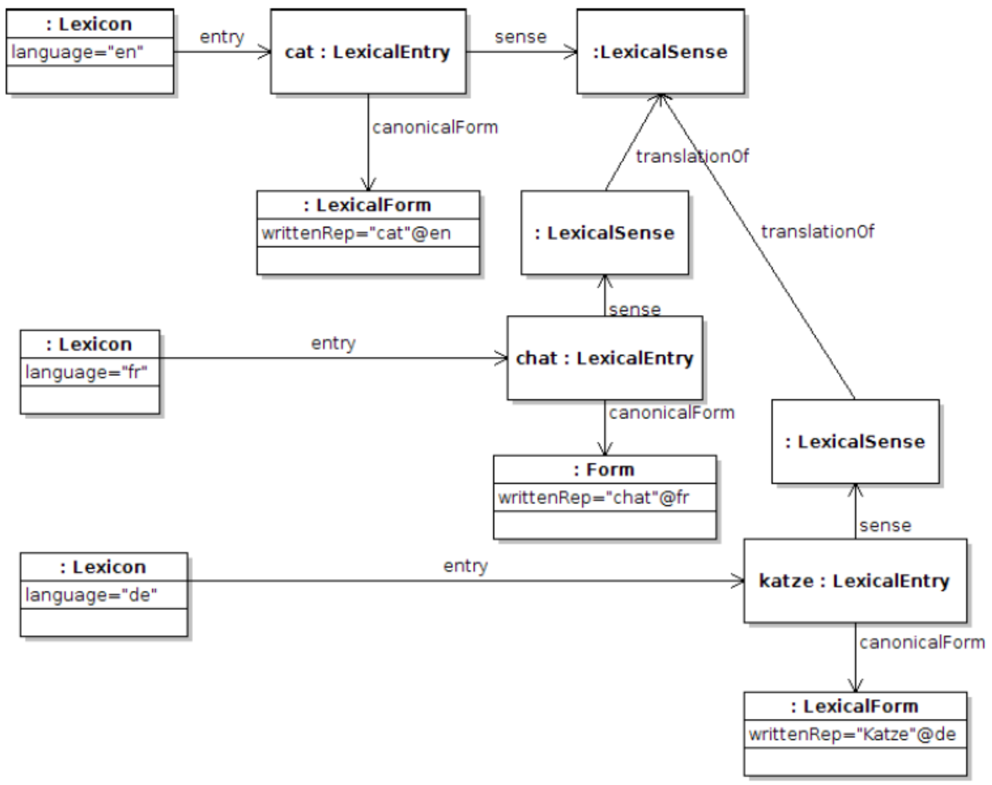


Figure 7.6: The *Lemon* model for multilingual ontologies to represent the class Cat (Source: [McCrae et al., 2012])

```
Popular Culture
     BT Culture
          NT n/a
               RT Globalization
               RT Literature
               RT Mass Media
               RT Media Literacy
               RT Films
                    UF Mass Culture (2004)
Mass Media
     BT n/a
          NT Films
          NT News Media
          NT Radio
               RT Advertising
               RT Propaganda
               RT Publications;
                    UF Multichannel Programing (1966 1980) (2004)
Propaganda
     BT Communication (Thought Transfer)
     BT Information Dissemination
          NT n/a
               RT Advertising
               RT Deception
               RT Mass Media
                    UF n/a
```

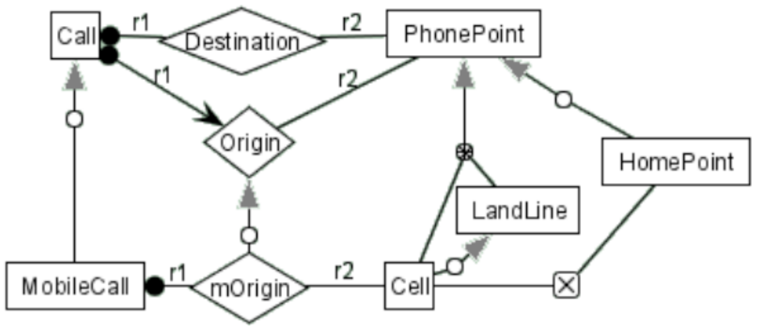


Figure 7.7: A small conceptual model in ICom (from its website `http://www.inf.unibz.it/~franconi/~icom` ; see [Fillottrani et al., 2012] for further details about the tool); blob: mandatory, open arrow: functional; square with star: disjoint complete, square with cross: disjoint, closed arrow (grey triangle): subsumption.

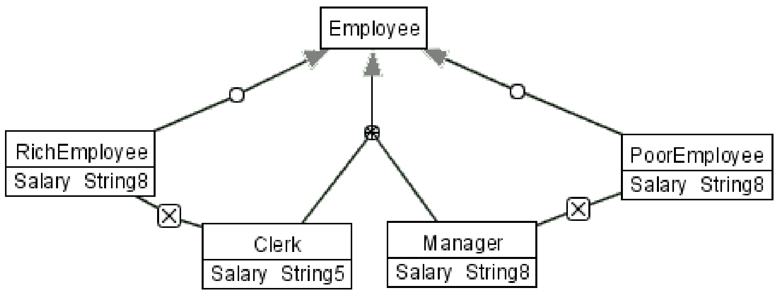


Figure 7.8: A small conceptual model in ICom (Source: `http://www.inf.unibz.it/~franconi/~icom`).

Exercise 43. The least common subsumer and most specific concept use non-standard reasoning services that helps with ontology development, and they are defined in terms of DL knowledge bases as follows.

Definition 7.1 (least common subsumer ([Peñaloza and Turhan, 2011])). *Let $\mathcal{L}$ be a Description Logic language, $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a knowledge base represented in DL $\mathcal{L}$ (an $\mathcal{L}$-KB). The* least common subsumer *(lcs) with respect to $\mathcal{T}$ of a collection of concepts $C_1, \dots, C_n$ is the $\mathcal{L}$-concept description C such that:*

1. *$C_i \sqsubseteq_{\mathcal{T}} C$ for all $1 \leq i \leq n$, and*
2. *for each $\mathcal{L}$-concept description D holds: if $C_i \sqsubseteq_{\mathcal{T}} D$ for all $1 \leq i \leq n$, then $C \sqsubseteq_{\mathcal{T}} D$.*

Definition 7.2 (most specific concept ([Peñaloza and Turhan, 2011])). *Let $\mathcal{L}$ be a Description Logic language, $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a knowledge base represented in DL $\mathcal{L}$ (an $\mathcal{L}$-KB). The* most specific concept *(msc) with respect to $\mathcal{K}$ of an individual from $\mathcal{A}$ is the $\mathcal{L}$-concept description C such that:*

1. *$\mathcal{K} \models C(a)$, and*
2. *for each $\mathcal{L}$-concept description D holds: $\mathcal{K} \models D(a)$ implies $C \sqsubseteq_{\mathcal{T}} D$.*

Describe in your own words what they do.

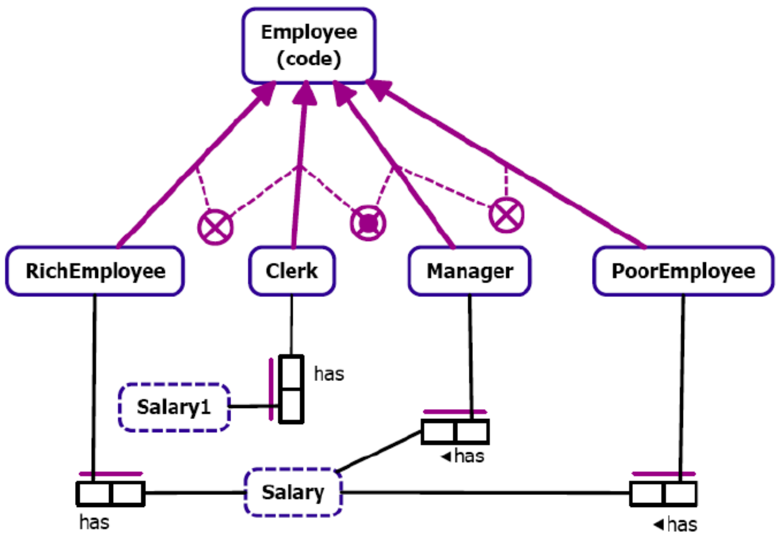


Figure 7.9: A small conceptual model in ORM2, similar to that in Figure 7.8.

Exercise 44. In what way(s) can data mining be useful in bottom-up ontology development? Your answer should include something about the following three aspects:

a. populating the TBox (learning classes and hierarchies, relationships, constraints),
b. populating the ABox (assertions about instances), and
c. possible substitutes or additions to the standard automated reasoning service (consistency checking, instance classification, etc.).

Exercise 45. You are an ontology consultant and have to advise the clients on ontology development for the following scenario. What would your advice be, assuming there are sufficient resources to realize it? Consider topics such as language, reasoning services, bottom-up, top-down, methods/methodologies.

> A pharmaceutical company is in the process of developing a drug to treat blood infections. There are about 100 candidate-chemicals in stock, categorised according to the BigPharmaChemicalsThesaurus, and they need to find out whether it meets their specification of the 'ideal' drug, codename DruTopiate, that has the required features to treat that disease (they already know that DruTopiate must have as part a benzene ring, must be water-soluble, smaller than 1 μm, etc). Instead of finding out by trial-and-error and test all 100 chemicals in the lab in costly experiments, they want to filter out candidate chemicals by automatic classification according to those DruTopiate features, and then experiment only with the few that match the desired properties. This *in silico* (on-the-computer) biomedical research is intended as a pilot study, and it is hoped that the successes obtained in related works, such as that of the protein phosphatases and ideal rubber molecules, can be achieved also in this case.

7.6 Literature and reference material

1. L. Lubyte, S. Tessaris. Automatic Extraction of Ontologies Wrapping Relational Data Sources. In *Proc. of the 20th International Conference on Database and Expert Systems Applications (DEXA 2009).*

2. Witte, R. Kappler, T. And Baker, C.J.O. Ontology design for biomedical text mining. In: *Semantic Web: revolutionizing knowledge discovery in the life sciences*, Baker, C.J.O., Cheung, H. (eds), Springer: New York, 2007, pp 281-313.
3. Dagobert Soergel, Boris Lauser, Anita Liang, Frehiwot Fisseha, Johannes Keizer and Stephen Katz. Reengineering thesauri for new applications: the AGROVOC example. *Journal of Digital Information* 4(4) (2004).
4. Keet, C.M. Bottom-up ontology development reusing semi-structured life sciences diagrams. *AFRICON'11 – Special Session on Robotics and Artificial Intelligence in Africa*, Livingstone, Zambia 13-15 September, 2011. IEEE Computer Society, pp1-6.
5. SKOS Core[12], SKOS Core guide[13], and the SKOS Core Vocabulary Specification[14].
6. OMG's Ontology definition metamodel with interactions between UML and OWL & RDF[15]

[12]http://www.w3.org/2004/02/skos/core
[13]http://www.w3.org/TR/swbp-skos-core-guide
[14]http://www.w3.org/TR/swbp-skos-core-spec
[15]http://www.omg.org/spec/ODM/1.0/

Part III

Advanced Topics

CHAPTER 8

Ontology-Based Data Access

The previous two blocks were rather theoretical in the sense that we have not seen many practical application infrastructures with ontologies. This is set to change in the upcoming two lectures of the "advanced topics". We shall look at both theoretical foundations of *ontology-based data access* (OBDA)—currently rebranded as ontology-based data management[1]—and one of its realisations, and you will set up an OBDA system yourselves during the labs.

There are several 'starting points' for introducing OBDA, depending on one's background. The short description is that it links an 'ontology' to lots of data in a relational database by means of a newly introduced mapping layer, which subsequently can be used for (automated reasoner-enhanced) 'intelligent' queries. The sneer quotes on 'ontology' have to do with the fact that, practically, the 'ontology' is a logic-based ($\pm$OWL 2 QL) simple conceptual data model. The sneer quotes on 'intelligent' refer to the fact that with the knowledge represented in the ontology/conceptual data model and an OBDA-enabled reasoner, one can pose more advanced queries to the database (in some cases) than with just a plain relational database and SQL.

In any case, we divert from the "we don't really care about expressivity and scalability" of the previous block to a setting that is driven by the need for scalability of ontology-driven information systems.

8.1 Introduction: Motivations

To motivate the need for some version of an OBDA system, we start with two perspectives: the (end-)user and the sysadmin.

A systems administrator's perspective

Organisations normally have multiple database to store and manage their data; e.g., the SMS for student, module, and grade management at UKZN, a database with employee data, the Moodle system for the university's modules' content management, and so on. Or take the Johannesburg public administration that wants to develop integrated services

[1]For an informal overview, you may like to read also Maurizio Lenzerini's blog post of the Association of Computing Machinery Special Interest Group on Management Of Data (ACM SIGMOD)'s blog at `http://wp.sigmod.org/?p=871`, May 14, 2013.

delivery and the separate databases of the individual services have to be integrated. Health information systems sometimes need to be kept separate for privacy reasons, yet at the same time, some cross-database queries have to be executed. Hence, the databases have to be connected in some way, and doing all that manually is a tedious, time-consuming task in any case. Moreover, you will have to know *how* the data is stored in the database, write (very) large queries, and there is no management for recurring queries.

Instead of knowing the structure of the database(s) by heart to construct such large queries, one can reduce the cognitive (over-)load by focusing on *what* is in the database, without having to care about if the class `Student` has a separate table or not, and whether it uses the full name `student` or perhaps some abbreviations, like `stud`. Provided the database was developed properly, there is a conceptual data model that has exactly the representation of what kind of data is stored in the database, but, traditionally, this is offline and shelved after implementation. However, this need not be the case, and OBDA (as well as OBDI—ontology-based data integration) fills this gap.

The case from the viewpoint of the user

Did you ever not want to bother knowing how the data is stored in a database, but simply want to know what kind of things are stored in the database at, say, the conceptual or ontological layer of knowledge? And did you ever not want to bother writing queries in SQL (or, in the context of the Semantic Web, SPARQL), but have a graphical point-and-click interface with which you can compose a query using that 'what layer' of knowledge and that the system generates automatically the SQL/SPARQL query for you, in the correct syntax? And all that not with a downloaded desktop application but in a Web browser? Frustrated with canned queries and pre-computed queries that limit your freedom to analyse the data? You don't want to keep on bothering the sysadmin for application layer updates to meet your whims and be dependent on whether she has time for your or not?

Several domain experts in genetics as well as in healthcare informatics, at least, wanted that and felt constrained in what they could do with their data, including at least pondering about, if not full desperation with, the so-called "write-only" databases. Especially in the biology and biomedical fields there has been much ontology development as well as generation of much data, in parallel, which somehow has to be linked up again.

The notion of *query by diagram* fills this gap. Of itself, this idea is not new [Catarci and Santucci, 1994, Bloesch and Halpin, 1996, Bloesch and Halpin, 1997], but now the technologies exist to realise it, even through a web interface and with reasoner-enabled querying. So, now one can do a sophisticated analysis of one's data and unlock new information from the database by using the ontology-based approach. In one experiment, this resulted in the users—scientists conducting *in silico* experiments—coming up with new queries not thought of asking before [Calvanese et al., 2010].

8.2 OBDA Architecture

The intuitive idea for solving the sysadmin issues is depicted in Figure 8.1: we add a "semantic layer" to a traditional database, or: we have a semantic layer and store information about all individuals in the knowledge base not in the OWL ABox but in external storage (a relational database) and create a new link between the OWL TBox and the data store.

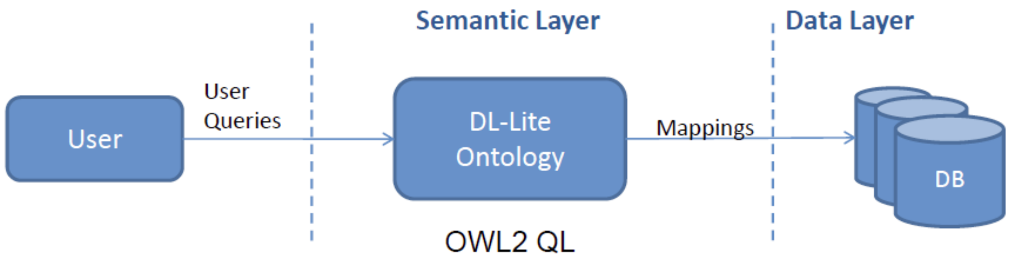


Figure 8.1: OBDA approach.

There are several approaches to realise this, principally from a databases perspective and from a knowledge representation perspective, or, informally, “a database with background knowledge in a logical theory” versus “a logical theory (‘ontology’) with lots of data in external storage”, which give rise to different formalisations due to their different starting points. Within the knowledge representation perspective, there are alternatives, too, both in theory [Calvanese et al., 2007, Kontchakov et al., 2010, Lutz et al., 2009] and then there are different tools for each component that makes up a realised OBDA system. For instance, one can choose less or no reasoning, such as the Virtuoso system[2], and an RDF triple store versus relational database technology to store the data. We shall take a look at the principal options, and subsequently consider in more detail the OBDA system developed at “La Sapienza” University in Rome and Free University of Bozen-Bolzano, Italy, which is described in [Calvanese et al., 2009]. Its principal ingredients are:

- Formal language: a language in the *DL-Lite* family, (roughly OWL 2 QL);
- OBDA-enabled reasoner: e.g., QuOnto [Acciarri et al., 2005] or Quest[3] [Rodríguez-Muro and Calvanese, 2012];
- Data storage: an RDBMS, e.g., Oracle, PostgreSQL, DB2;
- Developer interface: OWL ontology development environment, such as Protégé and an OBDA plugin [Rodriguez-Muro et al., 2008], to manage the mappings and data access, and a developer API facing toward the application to be developed;
- End-user interface: OBDA plugin for Protégé for SPARQL queries[4] and results [Rodriguez-Muro et al., 2008], and the wonder system for graphical querying [Calvanese et al., 2010][5].

This is shown schematically in Figure 8.2. Motivations as to why it is this configuration will be discussed during the lecture, though some of it can already be gleaned from the OWL 2 Profiles specification [Motik et al., 2009a] and Section 4.3.

8.3 Principal components

The theoretical details are quite involved, and during the two lectures exemplary excerpts of the key components will be described. There are two principal aspects to it:

[2]`http://virtuoso.openlinksw.com/`, used for, among others DBPedia

[3]`http://obda.inf.unibz.it/protege-plugin/quest/quest.html`

[4]SPARQL is a query language, alike SQL but then for querying OWL and RDF. It deserves more time to cover than we will do during the lecture. The W3C specification of SPARQL can be found at `http://www.w3.org/TR/rdf-sparql-query/`, and some RDF/SPARQL tools at `http://www.franz.com/products/allegrograph/` (AllegroGraph), `http://sourceforge.net/projects/sesame/` (Sesame), and `http://sites.wiwiss.fu-berlin.de/suhl/bizer/d2rq/` (D2RQ).

[5]Wonder with input from Santi Garcia-Vallvé (with the Evolutionary Genomics Group, ‘Rovira i Virgilli’ University, Tarragona, Spain) and Mark van Passel (with the Laboratory for Microbiology, Wageningen University and Research Centre, the Netherlands).

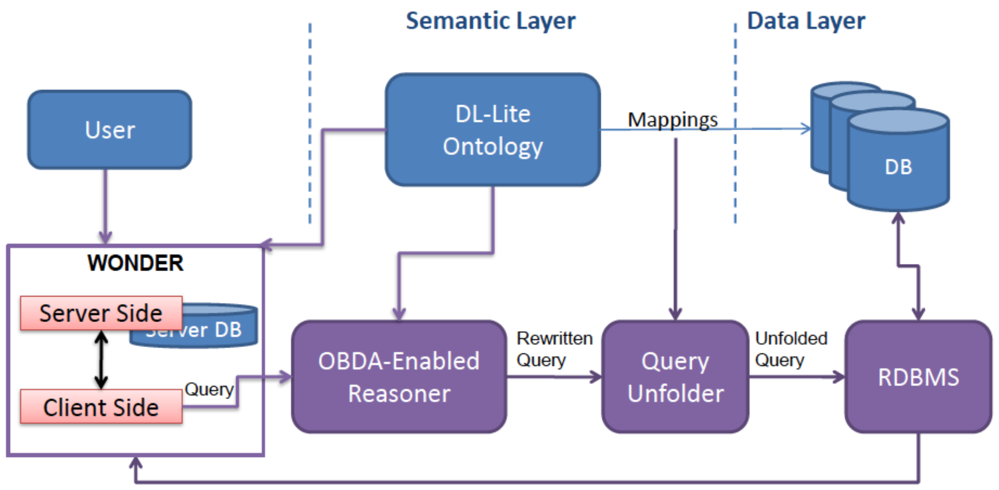


Figure 8.2: OBDA approach and some practical components (bottom-half): the relational database, mappings, an ontology, a reasoner, and a user interface (called WONDER) to hide the technicalities from the end-user.

1. Ontology-Based Data Access systems (static components):
   - An ontology language
   - A mapping language
   - The data
2. Query answering in Ontology-Based Data Access systems:
   - Reasoning over the TBox
   - Query rewriting
   - Query unfolding
   - Relational database technology

The mapping language and the items under 2 are new notions compared to the previous material in this module (in particular, Section 3) and the databases module (COMP306).

The ontology language

For the language, we remain, roughly, within the Semantic Web setting, and take a closer look at the OWL 2 QL profile and similar languages in the "DL-lite" family that is at the basis of OWL 2 QL, and *DL-Lite*$_{\mathcal{A}}$ in particular [Calvanese et al., 2007]. Most significantly, the trade-off between expressive power and computational complexity of the reasoning services leans strongly towards the scalability of reasoning (including query answering) over large amounts of data; or: one can't say a lot with either OWL 2 QL or *DL-Lite*$_{\mathcal{A}}$.

Syntax of *DL-Lite*$_{\mathcal{A}}$. As is common in DLs and OWL, we distinguish between (abstract) objects and (data) values. A *class expression* denotes a set of objects, a *datatype* denotes a set of values, an *object property* denotes a binary relationship between objects, and a *data property* denotes a binary relation between objects and values. We assume to have a set $\{T_1, \ldots, T_n\}$ of pairwise disjoint and unbounded datatypes, each denoting a set $val(T_i)$ of values (integers, strings, etc.), and $\top_d$ denotes the set of all values. Class expressions, C, and object property expressions, R, are formed according to the following syntax, where A denotes a class, P an object property, and U a data property:

$$C \longrightarrow A \mid \exists R \mid \delta(U), \qquad R \longrightarrow P \mid P^{-}.$$

Observe that $\exists R$ denotes an unqualified existential class expression, $\delta(U)$ denotes the domain of U, and P^- denotes the inverse of P.

A *DL-Lite*$_\mathcal{A}$ ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$, consists of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$, where the TBox is constituted by a set of *axioms* of the form

$$\begin{array}{cccc} C_1 \sqsubseteq C_2, & \rho(U) \sqsubseteq T_i, & R_1 \sqsubseteq R_2, & U_1 \sqsubseteq U_2, \\ (\mathsf{disj}\ C_1\ C_2), & & (\mathsf{disj}\ R_1\ R_2), & (\mathsf{disj}\ U_1\ U_2), \\ & & (\mathsf{funct}\ R), & (\mathsf{funct}\ U). \end{array}$$

The axioms in the first row denote *inclusions*, with $\rho(U)$ the range of U. The axioms in the second row denote *disjointness*; note that distinct datatypes are assumed to be disjoint. The axioms in the third row denote *functionality* (at most one) of an object property expression and of a data property expression, respectively. The ABox is constituted by a set of *assertions* of the form $A(a)$, $P(a, a')$, and $U(a, \ell)$, where a, a' are individuals denoting objects and ℓ is a literal (denoting a value). To ensure that *DL-Lite*$_\mathcal{A}$ maintains the computationally well-behaved computational properties of the *DL-Lite* family [Calvanese et al., 2007], the form of the TBox has to be restricted (as we have seen for OWL 2 QL in Chapter 3). In particular, object and data properties occurring in functionality assertions cannot be specialized, i.e., the cannot appear in the right hand side of an inclusion axiom.

Semantics of ***DL-Lite***$_\mathcal{A}$**.** The semantics of *DL-Lite*$_\mathcal{A}$ is based on first-order *interpretations* $\mathcal{I} = \mathcal{I}$, where $\Delta^\mathcal{I}$ is a nonempty interpretation domain, which is partitioned into a $\Delta^\mathcal{I}_O$ of objects, and a $\Delta^\mathcal{I}_V$ of values. The interpretation function $\cdot^\mathcal{I}$ maps each individual a to $a^\mathcal{I} \in \Delta^\mathcal{I}_O$, each class A to $A^\mathcal{I} \subseteq \Delta^\mathcal{I}_O$, each object property P to $P^\mathcal{I} \subseteq \Delta^\mathcal{I}_O \times \Delta^\mathcal{I}_O$, and each data property U to $U^\mathcal{I} \subseteq \Delta^\mathcal{I}_O \times \Delta^\mathcal{I}_V$, whereas each literal ℓ is interpreted as the value $\ell^\mathcal{I} = \mathit{val}(\ell)$, each datatype T_i as the set of values $T_i^\mathcal{I} = \mathit{val}(T_i)$, and $\top_d^\mathcal{I} = \Delta^\mathcal{I}_V$. The semantics of expressions:

$$(\exists R)^\mathcal{I} = \{o \mid \exists o'.\ (o, o') \in R^\mathcal{I}\}, \qquad (P^-)^\mathcal{I} = \{(o, o') \mid (o', o) \in P^\mathcal{I}\},$$

$$(\delta(U))^\mathcal{I} = \{o \mid \exists v.\ (o, v) \in U^\mathcal{I}\}, \qquad (\rho(U))^\mathcal{I} = \{v \mid \exists o.\ (o, v) \in U^\mathcal{I}\}.$$

Contrary to OWL, *DL-Lite*$_\mathcal{A}$ (and its implementation) adopts the *unique name assumption*, meaning that for every interpretation $\mathcal{I}$ and distinct individuals or values c_1, c_2, we have that $c_1^\mathcal{I} \neq c_2^\mathcal{I}$ (which is the norm in the database setting).

As for other DLs and OWL species, $\mathcal{I}$ *satisfies* $\alpha_1 \sqsubseteq \alpha_2$ if $\alpha_1^\mathcal{I} \subseteq \alpha_2^\mathcal{I}$, it satisfies $(\mathsf{disj}\ \alpha_1\ \alpha_2)$ if $\alpha_1^\mathcal{I} \cap \alpha_2^\mathcal{I} = \emptyset$, and it satisfies $(\mathsf{funct}\ S)$ if $S^\mathcal{I}$ is a function (that is, if $(o, z_1) \in S^\mathcal{I}$ and $(o, z_2) \in S^\mathcal{I}$, then $z_1 = z_2$)). $\mathcal{I}$ satisfies $A(a)$ if $a^\mathcal{I} \in A^\mathcal{I}$, it satisfies $P(a, a')$ if $(a^\mathcal{I}, a'^\mathcal{I}) \in P^\mathcal{I}$, and it satisfies $U(a, \ell)$ if $(a^\mathcal{I}, \mathit{val}(\ell)) \in U^\mathcal{I}$.

Mappings

Here, a few definitions and an example is included; the lecture slides and chapter literature contain further technical details and more examples of mappings.

Definition 8.1 (Mapping assertion between a database and a TBox)**.** *A mapping assertion between a database* $\mathcal{D}$ *and a TBox* $\mathcal{T}$ *has the form* $\Phi \rightsquigarrow \Psi$ *where*

- Φ *is an arbitrary SQL query of arity* $n > 0$ *over* $\mathcal{D}$*;*
- Ψ *is a conjunctive query over* $\mathcal{T}$ *of arity* $n' > 0$ *without non-distinguished variables, possibly involving variable terms.*

Definition 8.2 (Mapping assertion in $\mathcal{M}$ in an OBDA system)**.** *A mapping assertion between a database $\mathcal{D}$ and a TBox $\mathcal{T}$ in $\mathcal{M}$ has the form $\Phi(\vec{x}) \rightsquigarrow \Psi(\vec{t}, \vec{y})$ where*

- *Φ is an arbitrary SQL query of arity $n > 0$ over $\mathcal{D}$;*
- *Ψ is a conjunctive query over $\mathcal{T}$ of arity $n' > 0$ without non-distinguished variables;*
- *$\vec{x}, \vec{y}$ are variables with $\vec{y} \subseteq \vec{x}$;*
- *$\vec{t}$ are variable terms of the form $f(\vec{z})$, with $f \in \Lambda$ and $\vec{z} \subseteq \vec{x}$.*

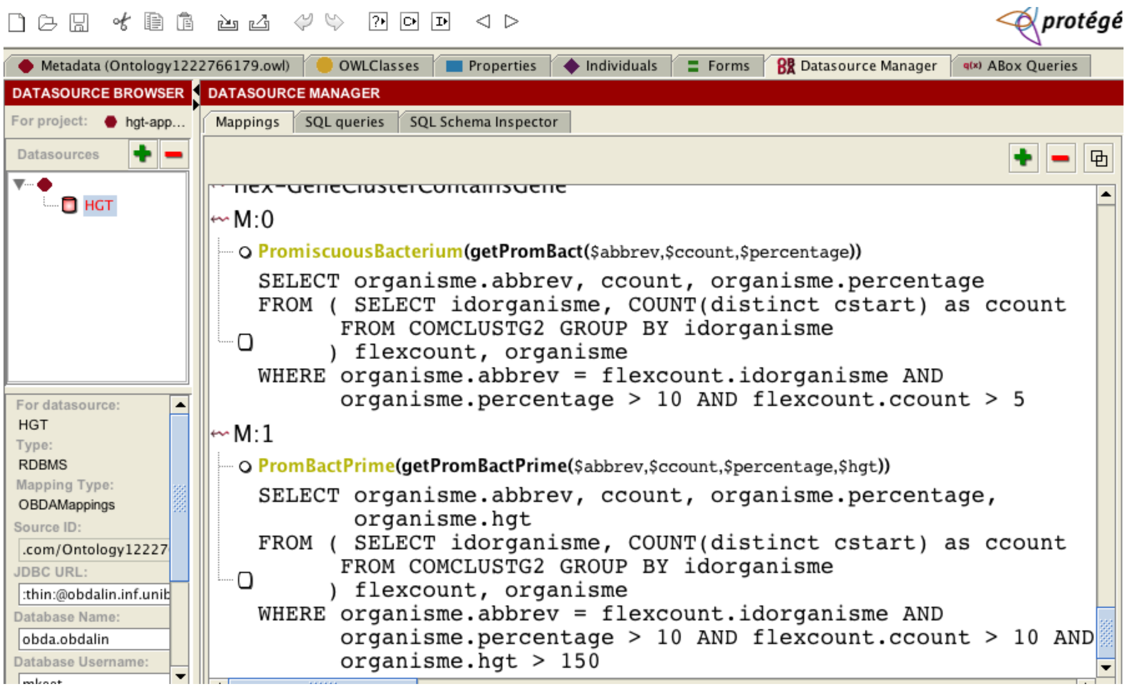
Concerning the the semantics of mappings, intuitively: $\mathcal{I}$ satisfies $\Phi \rightsquigarrow \Psi$ with respect to $\mathcal{D}$ if all facts obtained by evaluating Φ over $\mathcal{D}$ and then propagating answers to Ψ, hold in $\mathcal{I}$.

Definition 8.3 (Satisfaction of a mapping assertion with respect to a database)**.** *An interpretation $\mathcal{I}$ satisfies a mapping assertion $\Phi(\vec{x}) \rightsquigarrow \Psi(\vec{t}, \vec{y})$ in $\mathcal{M}$ with respect to a database $\mathcal{D}$, if for each tuple of values $\vec{v} \in Eval(\Phi, \mathcal{D})$, and for each ground atom in $\Psi[\vec{x}/\vec{v}]$, we have that:*

- *If the ground atom is $A(s)$, then $s^{\mathcal{I}} \in A^{\mathcal{I}}$;*
- *If the ground atom is $P(s_1, s_2)$, then $(s_1^{\mathcal{I}}, s_2^{\mathcal{I}}) \in P^{\mathcal{I}}$.*

(Note: $Eval(\Phi, \mathcal{D})$ denotes the result of evaluating Φ over $\mathcal{D}$, $\Psi[\vec{x}/\vec{v}]$ denotes Ψ where each x_i is substituted with v_i)

An example is shown in Figure 8.3 with the OBDA plugin for Protégé. There is an ontology that happens to have a class PromiscuousBacterium, among other things, and a relational database (HGT) with several of tables, such as `organisme` and `flexcount`. Now we have to link the two with a mapping, which means (i) constructing a database query such that it retrieves only the promiscuous bacteria, and (ii) solving the 'impedance mismatch' (recollect Chapter 7) with a functor so that the values returned by the database query become objects in the ontology, which is what getPromBact does. Informally, the functor can be considered as a URI building mechanism for individuals in the ontology taken from the database (theoretically, they are skolem functions).



For project: hgt-app...
HGT
For datasource: HGT
Type: RDBMS
Mapping Type: OBDAMappings
Source ID: .com/Ontology12227
JDBC URL: :thin:@obdalin.inf.unib
Database Name: obda.obdalin
Database Username:

M:0

PromiscuousBacterium(getPromBact($abbrev,$ccount,$percentage))

```
SELECT organisme.abbrev, ccount, organisme.percentage
FROM ( SELECT idorganisme, COUNT(distinct cstart) as ccount
       FROM COMCLUSTG2 GROUP BY idorganisme
      ) flexcount, organisme
WHERE organisme.abbrev = flexcount.idorganisme AND
      organisme.percentage > 10 AND flexcount.ccount > 5
```

M:1

PromBactPrime(getPromBactPrime($abbrev,$ccount,$percentage,$hgt))

```
SELECT organisme.abbrev, ccount, organisme.percentage,
       organisme.hgt
FROM ( SELECT idorganisme, COUNT(distinct cstart) as ccount
       FROM COMCLUSTG2 GROUP BY idorganisme
      ) flexcount, organisme
WHERE organisme.abbrev = flexcount.idorganisme AND
      organisme.percentage > 10 AND flexcount.ccount > 10 AND
      organisme.hgt > 150
```

Figure 8.3: An example of a mapping; see text for explanation. (Source: [Keet, 2010c])

Query answering

Recall the outline of the ADOLENA ontology from the exercise and Figure 6.8. A query could be

```
    q(x) :- Device(x), ameliorates(x,y), Paraplegia(y)
```

i.e., " retrieve the devices that ameliorate paraplegia". For this to work, we have to introduce three aspects. First, we need a computer-processable serialization of the query, a notion of what kind of queries we can pose, and a way how it will do the answering. The query language is SPARQL (see footnote 4). The kind of queries are (unions of) conjunctive queries. A conjunctive query (CQ) q over an ontology $\mathcal{O}$ is an expression of the form $q(\vec{x}) \leftarrow \exists \vec{y}. conj(\vec{x}, \vec{y})$, where $q(\vec{x})$ the head, $conj(\vec{x}, \vec{y})$ the body, the variables in $\vec{x}$ are distinguished variables and $\vec{y}$ the non-distinguished variables, and where $conj(\vec{x}, \vec{y})$ is a conjunction of atoms of the form $D(z)$, $S(z, z')$, $z = z'$, where D denotes a class or a datatype, S an object property or data property in $\mathcal{O}$, and z, z' are individuals or literals in $\mathcal{O}$ or variables in $\vec{x}$ or $\vec{y}$. Given an interpretation $\mathcal{I} = \mathcal{I}$, then $q^{\mathcal{I}}$ is the set of tuples of $\Delta^{\mathcal{I}}$ that, when assigned to the variables $\vec{x}$, make the formula $\exists \vec{y}. conj(\vec{x}, \vec{y})$ true in $\mathcal{I}$. The set $cert(q, \mathcal{O})$—certain answers to q over $\mathcal{O}$—is the set of tuples $\vec{a}$ of individuals or literals appearing in $\mathcal{O}$ such that $\vec{a}^{\mathcal{I}} \in q^{\mathcal{I}}$, for every model $\mathcal{I}$ of $\mathcal{O}$.

Regarding query answering, consider Figure 8.4, where q is our query, $\mathcal{T}$ the TBox (ontology or formal conceptual data model), and $\mathcal{A}$ the ABox (our instances, practically stored in the relational database). Somehow, this combines to produce the answers, using all components.

First, there is a "reformulation" (or: rewriting) step, which computes the perfect reformulation (rewriting), q_{pr}, of the original query q using the inclusion assertions of $\mathcal{T}$ so that we have a union of conjunctive queries. That is, it uses the knowledge of the ontology to come up with the 'real' query. For instance, recollect Figure 6.8 about the ontology of abilities and disabilities, then a query "retrieve all Devices that assistWith UpperLimbMobility", i.e.,

```
    q(x) :- Device(x), assistsWith(x,y), UpperLimbMobility(y)
```

or, in SPARQL notation:

```
    SELECT $device
       WHERE {$device rdf:type :Device.
             $device :assistsWith $y.
             $y rdf:type :UpperLimbMobility}
```

will traverse the hierarchy of devices until it finds those devices that have an object property declared with as range UpperLimbMobility. In this case, this is MotorisedWheelchair, hence, the 'real' query concerns only the retrieval of the motorised wheelchairs (not first retrieving all devices, and then making the selection).

Second, the "unfolding" step computes a new query, q_{unf}, by using the (split version of) the mappings that link the terms of the ontology to queries over the database.

Third, the "evaluation" step delegates the evaluation of q_{unf} to the relational DBMS managing the data, which subsequently returns the answer.

8.4 Examples

An example of a *DL-Lite*$_{\mathcal{A}}$ ontology about the European football league can be found in [Calvanese et al., 2009], pp270-274, and [Calvanese et al., 2010]'s HGT ontology and mappings are online[6]. An independent extension is described in [Calbimonte et al., 2010], where OBDA is used to access streaming data sources.

[6]`http://obda.inf.unibz.it/obdahgtdb/obdahgtdb.html`

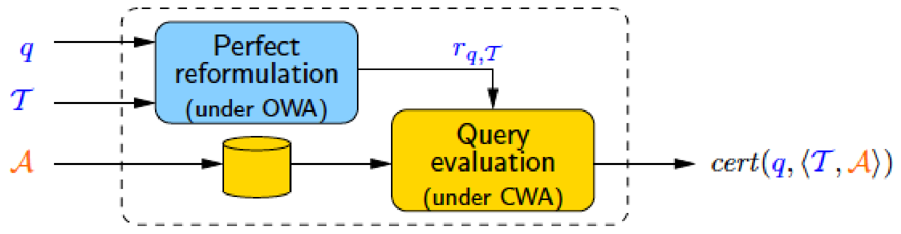

q
$\mathcal{T}$
$\mathcal{A}$
Perfect reformulation (under OWA)
$r_{q,\mathcal{T}}$
Query evaluation (under CWA)
$cert(q, \langle \mathcal{T}, \mathcal{A} \rangle)$

Figure 8.4: Intuition of the top-down approach to query answering.

More TBA.

8.5 Exercises

Exercise 46. You will set up an OBDA system with PostgreSQL, Quest, Protégé, and the OBDA Plugin for Protégé. Consult `https://babbage.inf.unibz.it/trac/obdapublic/wiki#Tutorials` for the files, step-wise explanations, and exercises. All tutorials together take about 3 day's work, so you may prefer to take a selection and distribute the tasks amongst your fellow students and present it to each other afterward (by the end of the week, the class has to have at least one working OBDA system).

Exercise 47. Work on your chosen mini-project.

8.6 Literature and reference material

1. Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Antonella Poggi, Mariano Rodriguez-Muro, and Riccardo Rosati. Ontologies and databases: The DL-Lite approach. In Sergio Tessaris and Enrico Franconi, editors, *Semantic Technologies for Information Systems - 5th Int. Reasoning Web Summer School (RW 2009)*, volume 5689 of Lecture Notes in Computer Science, pages 255-356. Springer, 2009.
2. Calvanese, D., Keet, C.M., Nutt, W., Rodríguez-Muro, M., Stefanoni, G. Web-based Graphical Querying of Databases through an Ontology: the WONDER System. *ACM Symposium on Applied Computing (ACM SAC'10)*, March 22-26 2010, Sierre, Switzerland.

CHAPTER 9

Extra topics

There are many topics in ontology engineering that deserve attention, which range from foundational issues, to engineering solely in the scope of the Semantic Web, to application areas with respect to peculiarities of a particular subject domain. For instance, ontology matching and alignment [Euzenat and Shvaiko, 2007] has a 'long' history (since the mid-1990s) whereas modularization is a current hot topic with a flurry of recent papers [Stuckenschmidt et al., 2009], the interaction of ontology with conceptual data models and reasoning should be, and work toward temporal ontologies is on the rise, whereas the social dimension of ontology engineering is inherent in the endeavour. Here I will briefly describe some of these topics, for which we may have time at the end of the module.

9.1 Uncertainty and vagueness

This advanced ontology engineering topic concerns how to cope with uncertainty and vagueness in ontology languages and their reasoners—and what we can gain from all the extra effort. At the time of writing, this elective topic is mainly focused on theory and research.

Consider, for instance, information retrieval: to which *degree* is a web site, a page, a text passage, an image, or a video segment relevant to the information need and an acceptable answer to what the user was searching for? Or in the context of ontology alignment, one would want to know (automatically) to which *degree* the focal concepts of two or more ontologies represent the same thing, or are *sufficiently* overlapping. In an electronic health record system, one may want to classify patients based on their symptoms, such as throwing up *often*, having a *high* blood pressure, and *yellowish* eye colour. Or the probability that a person is infected with HIV is 23%, that a minimum student pass rate for a module ought to be 35%, or the probability that birds fly. How can software agents do the negotiation for your holiday travel plans that are specified imprecisely, alike "I am looking for a package holiday of *preferably less than* 10000 ZAR, but really no more that 11500 ZAR , for *about* 12 days in a *cold* country that has snow"?

The main problem to solve, then, is what and how to incorporate such *vague* or *uncertain* knowledge in OWL and its reasoners. To clarify these two terms upfront:

- Uncertainty: statements are true or false, but *due to lack of knowledge* we can only estimate to which probability / possibility / necessity degree they are true or false;

- Vagueness: statements involve concepts for which *there is no exact definition* (such as tall, small, close, far, cheap, expensive), which are then true to some degree, taken from a truth space.

The two principal approaches regarding uncertainty and the Semantic Web are probabilistic and possibilistic languages, ontologies, and reasoning services, where the former way of dealing with uncertainty receives a lot more attention than the latter. The two principal approaches regarding vagueness and the semantic web are fuzzy and rough extensions, where fuzzy receives more attention compared to the rough approach.

None of the extant languages and automated reasoners that can cope with vague or uncertain knowledge have made it into mainstream Semantic Web tools yet. There was a W3C incubator group on uncertainty[1], but it remained at that. This has not stopped research in this area; on the contrary. There are two principal strands in these endeavours: one with respect to extending DL languages and its reasoners, such as Pronto[2] that combines the Pellet reasoner with a probabilistic extension and FuzzyDL[3] that is a reasoner for fuzzy $\mathcal{SHIF}(D)$, and another strand that uses different techniques underneath OWL, such as Bayesian networks for probabilistic ontologies (e.g., PR-OWL[4]), and Mixed Integer Logic Programming for fuzzy ontologies. Within the former approach, one can make a further distinction between extensions of tableaux algorithms and rewritings to a non-uncertain/non-vague standard OWL language so that one of the generic DL reasoners can be used. For each of these branches, there are differences as to which aspects of probabilistic/possibilistic/fuzzy/rough are actually included.

We shall not cover all such permutations in the lecture, but instead focus on general aspects of the languages and tools. A good introductory overview can be found in [Straccia, 2008] (which also has a very long list of references to start delving into the topics (you may skip the DLP section)). Depending on your background education, you may find the more technical overview [Lukasiewicz and Straccia, 2008] useful as well. To get an idea of one of the more recent results on rough DL-based ontologies, you might want to glance over the theory and experimentation [Jiang et al., 2009, Keet, 2010b, Keet, 2010c, Keet, 2011a]. Last, I assume you have a basic knowledge of probability theory and fuzzy sets; if there are many people who do not, I will adjust the lecture somewhat, but you are warmly advised to look it up before the lecture if you do not know about it.

Literature and reference material

1. Thomas Lukasiewicz and Umberto Straccia. 2008. Managing Uncertainty and Vagueness in Description Logics for the Semantic Web. *Journal of Web Semantics*, 6:291-308. NOTE: only the section on fuzzy ontologies
2. Umberto Straccia and Giulio Visco. DLMedia: an Ontology Mediated Multimedia Information Retrieval System. In: *Proceedings of the International Workshop on Uncertainty Reasoning for the Semantic Web (URSW-08)*, 2008. this is an example of an application that uses fuzzy DL
3. Keet, C.M. On the feasibility of Description Logic knowledge bases with rough concepts and vague instances. *23rd International Workshop on Description Logics (DL'10)*, 4-7 May 2010, Waterloo, Canada.

[1]`http://www.w3.org/2005/Incubator/urw3/`
[2]`http://pellet.owldl.com/pronto/`
[3]`http://gaia.isti.cnr.it/~straccia/software/fuzzyDL/fuzzyDL.html`
[4]`http://www.pr-owl.org/`

4. Keet, C.M. Rough Subsumption Reasoning with rOWL. *SAICSIT Annual Research Conference 2011 (SAICSIT'11)*, Cape Town, South Africa, October 3-5, 2011. ACM Conference Proceedings, 133-140.

9.2 Ontology-Driven Conceptual Data Modelling

Many databases and software applications have been and are being developed that have, or should have been, developed following a software or database development methodology, i.e., going from requirements analysis to conceptual analysis to design-level code and then to the actual implementation. The output of the conceptual analysis stage for software development is a *conceptual data model*, represented in EER, UML, or ORM. These languages are not equivalent, and, as with the different features in different DL languages, one can assess the expressiveness of those languages, formalise them, and reason over them [Artale et al., 2007a, Berardi et al., 2005, Calvanese et al., 1998b, Fillottrani et al., 2012, Keet, 2009b, Queralt and Teniente, 2008]. The formalisation and subsequent automated reasoning improves the quality of a conceptual data model further and prevents software bugs in case inconsistencies were found in the model.

In addition, one can then also extend the language in a controlled and precise way so as to cater for, perhaps less common, features (temporal, spatial etc.). For instance, one can specify and refine UML's aggregation association or part-whole relation by availing of advances in Ontology, and make it applicable to conceptual data modeling [Artale et al., 2008, Guizzardi, 2005, Keet and Artale, 2008], which then provides scientific arguments and explain why, e.g., a protein chain's Residue's Coordinates are not part of the residue (as in [Bornberg-Bauer and Paton, 2002]), but an attribute that describes its location, and distinguish spatial containment from structural parthood [Keet and Artale, 2008]. In addition, as was mentioned already in Section 1.3, one can use an ontology to generate multiple conceptual data models where a subset of the classes and axioms from the ontology can be reused to construct a conceptual data model [El-Ghalayini et al., 2006, Jarrar et al., 2003, Sugumaran and Storey, 2006], thereby improving their quality and ensuring interoperability upfront.

These three scenarios with Ontology-Driven Conceptual Data Modelling can be integrated in the regular software and database development methodologies, as is depicted in Figure 9.1. An overview, examples, and its application for biological data analysis is described in [Keet, 2013].

Literature and reference material

1. D. Berardi, D. Calvanese, and G. De Giacomo. Reasoning on UML class diagrams. *Artificial Intelligence*, 168(1-2):70-118, 2005.
2. Giancarlo Guizzardi. *Ontological Foundations for Structural Conceptual Models.* Phd thesis, University of Twente, The Netherlands. Telematica Instituut Fundamental Research Series No. 15, 2005.
3. C.M. Keet. Ontology-driven formal conceptual data modeling for biological data analysis. In Mourad Elloumi and Albert Y. Zomaya, editors, *Biological Knowledge Discovery Handbook: Preprocessing, Mining and Postprocessing of Biological Data.* Wiley, 2012. in press

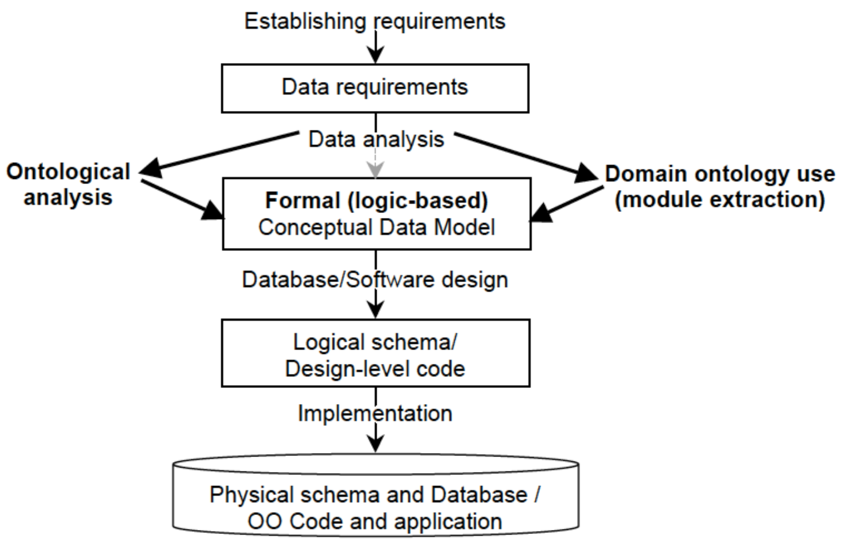

Establishing requirements
Data requirements
Data analysis
Ontological analysis
Domain ontology use (module extraction)
Formal (logic-based) Conceptual Data Model
Database/Software design
Logical schema/ Design-level code
Implementation
Physical schema and Database / OO Code and application

Figure 9.1: The so-called traditional 'waterfall' methodology, augmented with ontological analysis considering aspects from Ontology, Artificial Intelligence (more precisely: knowledge representation and reasoning), and ontologies, depicted in boldface.

9.3 Time and Temporal Ontologies

There are requests for including a temporal dimension in OWL; for instance, you can check the annotations in the OWL files of BFO and DOLCE (or, more conveniently, search for "time" in[5]) where they mention temporality that cannot be represented in OWL, or SNOMED CT[6] concepts like "Biopsy, *planned*" and "Concussion with loss of consciousness for *less than one hour*" where the loss of consciousness still can be before or after the concussion, or a business rule alike 'RentalCar must be returned *before* Deposit is reimbursed' or the symptom HairLoss *during* the treatment Chemotherapy, and Butterfly is a *transformation of* Caterpillar.

Unfortunately, there is no single (computational) solution to solve all these examples at once. Thus far, it is a bit of a patchwork of various theories and some technologies, with, among many aspects, the Allen's interval algebra [Allen, 1983] with qualitative temporal relations (such as before and during), Linear Temporal Logics (LTL) and Computational Tree Logics (CTL, with branching time), and a W3C Working draft of a time ontology[7] with more explanation in [Hobbs and Pan, 2004], among other things.

We will look at some motivations for temporal ontologies first, proceed to a temporal DL, $\mathcal{DLR}_{\mathcal{US}}$, and finally look at two modelling issues it helps solving.

9.3.1 Why temporal ontologies?

There are two principal parts to answering this question: because of *what* we want to represent and *what* inferencing we want to do with it.

The things to represent

Quite common time aspects in conceptual data modelling for information systems are the requirements to record actual dates and intervals and calendar calculations. This

[5]`http://www.meteck.org/files/swt/DolceliteBFOinDLandMSyntax.pdf`
[6]`http://www.ihtsdo.org/snomed-ct/`
[7]`http://www.w3.org/TR/owl-time/`

is not particularly relevant for a domain ontology, but it would be useful to have an ontology about such things so that the applications use the same notions and, hence, are interoperable in that regard.

In Chapter 6 we have seen BFO and the RO and the intention by its developers to add the *precedes* and *immediately precedes* relations to the OBO Foundry ontologies, which could not be done other than for annotation purposes. There are more such established qualitative temporal relations, also known as the 'Allen relations' or 'Allen's interval algebra', after the author who gave a first systematic and formal account of them [Allen, 1983], which comprise relations such as before, after, during, while, and meet[8]. Some might say they are all the temporal relations we need, but one may wish to be more specific in specialised subject domains, such as a *transformation_of* and *developed_from* in developmental biology.

A third aspect of the kinds of things we want to do, are temporalising classes and temporalising relations. The former is well-known in databases as 'object migration'; e.g., an active project evolves to a completed project, and each divorcee in the census database must have been married before. The latter, relation migration, is a new addition [Keet and Artale, 2010], which follows the idea of temporal classes, but then applied to n-ary tuples (with $n \geq 2$); e.g. 'during x's lifetime, it always has y as part' and 'every passenger that boards the plane must have checked in before departure of that flight'.

More comprehensive, and real, examples, can be found in, among others, [Artale et al., 2008, Keet, 2009a, Keet and Artale, 2010, Schulz et al., 2009], although this is not to say that all ontologies have, or ought have, a temporal component to represent the subject domain as accurately as possible.

Temporal reasoning services

As with a-temporal ontologies, one would want to have the same ones for temporal ontologies, such as satisfiability checking, subsumption reasoning, and classification. Querying temporal knowledge bases can also be of interest; to retrieve the answer to, e.g., "Who was the South African president after Nelson Mandela?". Logical implications are a bit more involved, though; e.g. given $B \sqsubseteq A$, then it must be the case that objects 'active' (alive) in B must be active in A and, e.g., to come up for promotion to become a company's manager (B), one must first exist as an employee (A) of that company.

There is a range of other examples which involve time in some way, and which have been solved and implemented already, such as reasoning with a calendar hierarchy and across calendars and finding a solution satisfying a set of constraints for scheduling the lecture hours of a study programme.

A few open issues

There are many issues that are being investigated. On the one hand, there are the modelling issues in ontology development and figuring out what temporal features they actually require in a temporal logic, the interaction between temporal logic and temporal databases (e.g., temporal OBDA), and further investigation into the interaction between temporal DLs with temporal conceptual data modelling. This, in turn, requires one to look into the computational properties of various fragments of expressive temporal logics. More fundamental issues have to do with making choices regarding linear time vs. branching time and endurantism vs. perdurantism.

[8] A quick introduction at `http://en.wikipedia.org/wiki/Allen's_Interval_Algebra`

9.3.2 Temporal DLs

If one assumes that recent advances in temporal Description Logics may have the highest chance of making it into a temporal OWL (tOWL)—although there are no proof-of-concept temporal DL modelling tools or reasoners yet—then the following is 'on offer'. A very expressive (undecidable) DL language is $\mathcal{DLR}_{\mathcal{US}}$ (with the $\mathcal{U}$ntil and $\mathcal{S}$ince operators), which already has been used for temporal conceptual data modelling [Artale et al., 2007c] and for representing essential and immutable parts and wholes [Artale et al., 2008]. A much less expressive language is TDL-Lite [Artale et al., 2007b], which is a member of the DL-Lite family of DL languages (of which one is the basis for OWL 2 QL); but these first results are theoretical, hence there is no "tOWL-lite" yet. It is already known that $\mathcal{EL}^{++}$ (the basis for OWL 2 EL) does not keep the nice computational properties when extended with LTL, and results with $\mathcal{EL}^{++}$ with CTL are not out yet. If you are really interested in the topic, you may want to have a look at a recent survey [Lutz et al., 2008] or take a broader scope with any of the four chapters from the KR handbook [van Harmelen et al., 2008] that cover temporal KR&R, situation calculus, event calculus, and temporal action logics, or the Handbook of temporal reasoning in artificial intelligence [Euzenat and Montanari, 2005]. During the lecture, we will take a look at $\mathcal{DLR}_{\mathcal{US}}$ and what it has been used for.

The $\mathcal{DLR}_{\mathcal{US}}$ temporal DL

$\mathcal{DLR}_{\mathcal{US}}$ [Artale et al., 2002] combines the propositional temporal logic with *Since* and *Until* operators with the a-temporal DL $\mathcal{DLR}$ [Calvanese and De Giacomo, 2003] and can be regarded as an expressive fragment of the first-order temporal logic $L^{\{\textbf{since, until}\}}$ [Chomicki and Toman, 1998, Hodgkinson et al., 1999, Gabbay et al., 2003].

As with other $\mathcal{DLR}$s, the basic syntactical types of $\mathcal{DLR}_{\mathcal{US}}$ are *classes* and n-ary *relations* ($n \geq 2$). Starting from a set of *atomic classes* (denoted by *CN*), a set of *atomic relations* (denoted by *RN*), and a set of *role symbols* (denoted by U), we can define complex class and relationship expressions (see upper part of Figure 9.2), where the restriction that binary constructors ($\sqcap, \sqcup, \mathcal{U}, \mathcal{S}$) are applied to relations of the same arity, i, j, k, n are natural numbers, $i \leq n$, j does not exceed the arity of R. All the Boolean constructors are available for both class and relation expressions. The selection expression $U_i/n : C$ denotes an n-ary relation whose i-th argument ($i \leq n$), named U_i, is of type C. (If it is clear from the context, we omit n and write $(U_i\!:\!C)$.) The projection expression $\exists^{\lessgtr k}[U_j]R$ is a generalisation with cardinalities of the projection operator over argument U_j of relation R; the classical projection is $\exists^{\geq 1}[U_j]R$.

The model-theoretic semantics of $\mathcal{DLR}_{\mathcal{US}}$ assumes a flow of time $\mathcal{T} = \langle \mathcal{T}_p, < \rangle$, where $\mathcal{T}_p$ is a set of time points and $<$ a binary precedence relation on $\mathcal{T}_p$, assumed to be isomorphic to $\langle \mathbb{Z}, < \rangle$. The language of $\mathcal{DLR}_{\mathcal{US}}$ is interpreted in *temporal models* over $\mathcal{T}$, which are triples of the form $\mathcal{I} \doteq \langle \mathcal{T}, \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}(t)} \rangle$, where $\Delta^{\mathcal{I}}$ is non-empty set of objects (the *domain* of $\mathcal{I}$) and $\cdot^{\mathcal{I}(t)}$ an *interpretation function*. Since the domain, $\Delta^{\mathcal{I}}$, is time independent, we assume here the so called *constant domain assumption* with *rigid designator*—i.e., an instance is *always* present in the interpretation domain and it identifies the same instance at different points in time. The interpretation function is such that, for every $t \in \mathcal{T}$ (a shortcut for $t \in \mathcal{T}_p$), every class C, and every n-ary relation R, we have $C^{\mathcal{I}(t)} \subseteq \Delta^{\mathcal{I}}$ and $R^{\mathcal{I}(t)} \subseteq (\Delta^{\mathcal{I}})^n$. The semantics of class and relation expressions is defined in the lower part of Fig. 9.2, where $(u, v) = \{w \in \mathcal{T} \mid u < w < v\}$. For classes, the temporal operators $\Diamond^+$ (some time in the future), $\oplus$ (at the next moment), and their past counterparts can be defined via $\mathcal{U}$ and $\mathcal{S}$: $\Diamond^+ C \equiv \top \,\mathcal{U}\, C$, $\oplus C \equiv \bot \,\mathcal{U}\, C$, etc. The operators $\Box^+$ (always in the future) and $\Box^-$ (always in the past) are the duals of $\Diamond^+$ (some time in the future)

$$\begin{array}{rcl}
C & \to & \top \mid \bot \mid CN \mid \neg C \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid \exists^{\leqslant k}[U_j]R \mid \\
 & & \Diamond^+ C \mid \Diamond^- C \mid \Box^+ C \mid \Box^- C \mid \oplus C \mid \ominus C \mid C_1 \mathcal{U} C_2 \mid C_1 \mathcal{S} C_2 \\
R & \to & \top_n \mid RN \mid \neg R \mid R_1 \sqcap R_2 \mid R_1 \sqcup R_2 \mid U_i/n : C \mid \\
 & & \Diamond^+ R \mid \Diamond^- R \mid \Box^+ R \mid \Box^- R \mid \oplus R \mid \ominus R \mid R_1 \mathcal{U} R_2 \mid R_1 \mathcal{S} R_2
\end{array}$$

$$\begin{array}{rcl}
\top^{\mathcal{I}(t)} & = & \Delta^{\mathcal{I}}; \\
\bot^{\mathcal{I}(t)} & = & \emptyset; \\
CN^{\mathcal{I}(t)} & \subseteq & \top^{\mathcal{I}(t)}; \\
(\neg C)^{\mathcal{I}(t)} & = & \top^{\mathcal{I}(t)} \setminus C^{\mathcal{I}(t)}; \\
(C_1 \sqcap C_2)^{\mathcal{I}(t)} & = & C_1^{\mathcal{I}(t)} \cap C_2^{\mathcal{I}(t)}; \\
(C_1 \sqcup C_2)^{\mathcal{I}(t)} & = & C_1^{\mathcal{I}(t)} \cup C_2^{\mathcal{I}(t)}; \\
(\exists^{\leqslant k}[U_j]R)^{\mathcal{I}(t)} & = & \{\, d \in \top^{\mathcal{I}(t)} \mid \sharp\{\langle d_1, \ldots, d_n \rangle \in R^{\mathcal{I}(t)} \mid d_j = d\} \leqslant k\}; \\
(C_1 \mathcal{U} C_2)^{\mathcal{I}(t)} & = & \{\, d \in \top^{\mathcal{I}(t)} \mid \exists v > t.(d \in C_2^{\mathcal{I}(v)} \wedge \forall w \in (t, v). d \in C_1^{\mathcal{I}(w)})\}; \\
(C_1 \mathcal{S} C_2)^{\mathcal{I}(t)} & = & \{\, d \in \top^{\mathcal{I}(t)} \mid \exists v < t.(d \in C_2^{\mathcal{I}(v)} \wedge \forall w \in (v, t). d \in C_1^{\mathcal{I}(w)})\}; \\
(\top_n)^{\mathcal{I}(t)} & \subseteq & (\Delta^{\mathcal{I}})^n; \\
RN^{\mathcal{I}(t)} & \subseteq & (\top_n)^{\mathcal{I}(t)}; \\
(\neg R)^{\mathcal{I}(t)} & = & (\top_n)^{\mathcal{I}(t)} \setminus R^{\mathcal{I}(t)}; \\
(R_1 \sqcap R_2)^{\mathcal{I}(t)} & = & R_1^{\mathcal{I}(t)} \cap R_2^{\mathcal{I}(t)}; \\
(R_1 \sqcup R_2)^{\mathcal{I}(t)} & = & R_1^{\mathcal{I}(t)} \cup R_2^{\mathcal{I}(t)}; \\
(U_i/n : C)^{\mathcal{I}(t)} & = & \{\langle d_1, \ldots, d_n \rangle \in (\top_n)^{\mathcal{I}(t)} \mid d_i \in C^{\mathcal{I}(t)}\}; \\
(R_1 \mathcal{U} R_2)^{\mathcal{I}(t)} & = & \{\langle d_1, \ldots, d_n \rangle \in (\top_n)^{\mathcal{I}(t)} \mid \exists v > t.(\langle d_1, \ldots, d_n \rangle \in R_2^{\mathcal{I}(v)} \wedge \forall w \in (t, v).\, \langle d_1, \ldots, d_n \rangle \in R_1^{\mathcal{I}(w)})\}; \\
(R_1 \mathcal{S} R_2)^{\mathcal{I}(t)} & = & \{\, \langle d_1, \ldots, d_n \rangle \in (\top_n)^{\mathcal{I}(t)} \mid \exists v < t.(\langle d_1, \ldots, d_n \rangle \in R_2^{\mathcal{I}(v)} \wedge \forall w \in (v, t).\, \langle d_1, \ldots, d_n \rangle \in R_1^{\mathcal{I}(w)})\}; \\
(\Diamond^+ R)^{\mathcal{I}(t)} & = & \{\langle d_1, \ldots, d_n \rangle \in (\top_n)^{\mathcal{I}(t)} \mid \exists v > t.\, \langle d_1, \ldots, d_n \rangle \in R^{\mathcal{I}(v)}\}; \\
(\oplus R)^{\mathcal{I}(t)} & = & \{\langle d_1, \ldots, d_n \rangle \in (\top_n)^{\mathcal{I}(t)} \mid \langle d_1, \ldots, d_n \rangle \in R^{\mathcal{I}(t+1)}\}; \\
(\Diamond^- R)^{\mathcal{I}(t)} & = & \{\langle d_1, \ldots, d_n \rangle \in (\top_n)^{\mathcal{I}(t)} \mid \exists v < t.\, \langle d_1, \ldots, d_n \rangle \in R^{\mathcal{I}(v)}\}; \\
(\ominus R)^{\mathcal{I}(t)} & = & \{\langle d_1, \ldots, d_n \rangle \in (\top_n)^{\mathcal{I}(t)} \mid \langle d_1, \ldots, d_n \rangle \in R^{\mathcal{I}(t-1)}\}.
\end{array}$$

Figure 9.2: Syntax and semantics of $\mathcal{DLR}_{\mathcal{US}}$.

and $\Diamond^-$ (some time in the past), respectively, i.e., $\Box^+ C \equiv \neg\Diamond^+\neg C$ and $\Box^- C \equiv \neg\Diamond^-\neg C$, for both classes and relations. The operators $\Diamond^*$ (at some moment) and its dual $\Box^*$ (at all moments) can be defined for both classes and relations as $\Diamond^* C \equiv C \sqcup \Diamond^+ C \sqcup \Diamond^- C$ and $\Box^* C \equiv C \sqcap \Box^+ C \sqcap \Box^- C$, respectively.

A $\mathcal{DLR}_{\mathcal{US}}$ *knowledge base* is a finite set Σ of $\mathcal{DLR}_{\mathcal{US}}$ axioms of the form $C_1 \sqsubseteq C_2$ and $R_1 \sqsubseteq R_2$, with R_1 and R_2 being relations of the same arity. An interpretation $\mathcal{I}$ satisfies $C_1 \sqsubseteq C_2$ ($R_1 \sqsubseteq R_2$) if and only if the interpretation of C_1 (R_1) is included in the interpretation of C_2 (R_2) at *all time*, i.e., $C_1^{\mathcal{I}(t)} \subseteq C_2^{\mathcal{I}(t)}$ ($R_1^{\mathcal{I}(t)} \subseteq R_2^{\mathcal{I}(t)}$), for all $t \in \mathcal{T}$. Thus, $\mathcal{DLR}_{\mathcal{US}}$ axioms have a global reading. To see examples on how a $\mathcal{DLR}_{\mathcal{US}}$ knowledge base looks like we refer to the following sections where examples are provided.

Modelling

We shall look into two examples in more detail. On the one hand, they demonstrate how temporal issues can permeate both a specific domain as well as domain-independent issues, and, on the other hand, that we have to start *combining* different aspects from previous lectures so as to arrive at the solution.

The first one has to do with solving how to represent the “Assuming boxers must have their own hands and boxers are humans, is Hand part of Boxer in the same way as Brain is part of Human?" that we have encountered in Section 6.2. Recasting this problem into the temporal dimension, we can encode it in $\mathcal{DLR}_{\mathcal{US}}$ and prove the correctness of the intended behaviour [Artale et al., 2008].

The second example reconsiders the suitable relations for bio-ontologies and RO's *transformation_of* in particular. To capture that more accurately and precisely, we apply some of the OntoClean foundations and combine that with the notion of so-called status classes from temporal conceptual data modeling and its formal characterisation with $\mathcal{DLR}_{\mathcal{US}}$ [Keet, 2009a].

Literature and reference material

Suggested readings:

1. Alessandro Artale, Christine Parent, and Stefano Spaccapietra. Evolving objects in temporal information systems. *Annals of Mathematics and Artificial Intelligence (AMAI)*, 50:5-38, 2007. NOTE: only the $\mathcal{DLR}_{\mathcal{US}}$ part, the rest is optional.
2. Keet, C.M. and Artale, A. A basic characterization of relation migration. International Workshop on Fact-Oriented Modeling (ORM'10), Crete, Greece, October 27-29, 2010. Meersman, R. et al. (Eds.), *OTM Workshops*, Springer, LNCS 6428, 484-493.
3. Keet, C.M. Constraints for representing transforming entities in bio-ontologies. *11th Congress of the Italian Association for Artificial Intelligence (AI*IA 2009).* R. Serra and R. Cucchiara (Eds.), Reggio Emilia, Italy, Dec. 9-12, 2009. Springer-Verlag LNAI 5883, 11-20.
4. J. R. Hobbs and F. Pan. An ontology of time for the semantic web. *ACM Transactions on Asian Language Processing (TALIP): Special issue on Temporal Information Processing*, 3(1):6685, 2004.

Optional readings (more technical details, in descending order of being optional):

1. Artale, A., Guarino, N., and Keet, C.M. Formalising temporal constraints on part-whole relations. *11th International Conference on Principles of Knowledge Representation and Reasoning (KR'08).* Gerhard Brewka, Jerome Lang (Eds.) AAAI Press, pp 673-683. Sydney, Australia, September 16-19, 2008.
2. Alessandro Artale, Enrico Franconi, Frank Wolter and Michael Zakharyaschev. A Temporal Description Logic for Reasoning over Conceptual Schemas and Queries. In: *Proceedings of the 8th European Conference on Logics in Artificial Intelligence (JELIA'02)*, Cosenza, Italy, September 2002. LNAI, Springer-Verlag.
3. Alessandro Artale, Roman Kontchakov, Carsten Lutz, Frank Wolter and Michael Zakharyaschev. Temporalising Tractable Description Logics. In: *Proc. of the 14th International Symposium on Temporal Representation and Reasoning (TIME'07)*, Alicante, June 2007.
4. Franz Baader, Silvio Ghilardi, and Carsten Lutz. LTL over Description Logic Axioms. In: *Proceedings of the 11th International Conference on Principles of Knowledge Representation and Reasoning (KR2008)*, 2008. AAAI Press. Sydney, Australia, September 16-19, 2008.

9.4 Challenges in representation and reasoning over ontologies

The challenges for Semantic Web Technologies (SWT) in general (and for bio-ontologies in particular) are quite diverse, of which some concern the SWT proper and others are by its designers—and W3C core activities on standardization—considered outside their responsibility but still need to be done. Currently, for the software aspects, the onus is put on the software developers and industry to pick up on the proof-of-concept and working-prototype tools that have come out of academia and to transform them into the industry-grade quality that a widespread adoption of SWT requires. Although this aspect should not be ignored, we shall focus on the language and reasoning limitations during the lecture.

In addition to the language and corresponding reasoning limitations that passed the revue in the lectures on OWL, there are language limitations discussed and illustrated at length in various papers, e.g., [Schulz et al., 2009], where it might well be that extensions like the ones about uncertainty, vagueness and time can ameliorate or perhaps even solve the problem. Some of the issues outlined by Schultz and coauthors [Schulz et al., 2009] are modelling pitfalls, whereas others are real challenges that can be approximated to a greater or lesser extent. We shall look at several representation issues that go beyond the earlier examples of SNOMED CT's "brain concussion without loss of consciousness"; e.g. how would you represent in an ontology that *in most but not all cases* hepatitis has as symptom fever, or how would you formalize the defined concept "Drug abuse *prevention*", and (provided you are convinced it should be represented in an ontology) that the world-wide *prevalence* of diabetes mellitus is 2.8%?

One has to note, however, the initial outcome of a survey conducted with ontology developers [Alberts et al., 2008]: there were discrepancies between the language features that were actually used in the ontology and the perceived requirements for language features selected by the survey respondents. This goes in both directions, i.e., where more features were requested than have been used in the ontology the survey respondendts were involved in and more features were used than were requested. Given that selecting an ontology language is important for all four other design parameters (see Section 5.3 and Figure 5.6), it deserves further investigation how to overcome this mismatch.

Concerning challenges for automated reasoning, we shall look at two of the nine identified required reasoning scenarios [Keet et al., 2007], being the "model checking (violation)" and "finding gaps in an ontology and discovering new relations", thereby reiterating that it is the life scientists' high-level goal-driven approach and desire to use OWL ontologies with reasoning services to, ultimately, *discover novel information about nature*. You might find it of interest to read about the feedback[9] received from the SWT developers upon presenting that paper: some requirements are met in the meantime and new useful reasoning services were presented.

Literature and reference material

1. Stefan Schulz, Holger Stenzhorn, Martin Boekers, and Barry Smith. Strengths and limitations of formal ontologies in the biomedical domain. *Electronic Journal of Communication, Information and Innovation in Health (Special Issue on Ontologies, Semantic Web and Health)*, 3(1):31-45, 2009.

[9] `http://keet.wordpress.com/2007/06/11/reasoning-requirements-for-bio-ontologies-the-harvest-from-owled-dl-2007/`

2. C. Maria Keet, Marco Roos, and M. Scott Marshall. A survey of requirements for automated reasoning services for bio-ontologies in OWL. In *Proceedings of the 3rd Workshop on OWL: Experiences and Directions (OWLED 2007)*, volume 258 of CEUR-WS, 2007. 6-7 June 2007, Innsbruck, Austria.

9.5 Social aspects

Some of these issues in ontology development also have to do with the tension between the "montagues" and the "capulets". That is, social aspects, which are lightly described in [Goble and Wroe, 2004], which is a write-up of Goble's presentation about the montagues and capulets[10] at the SOFG'04 meeting[11]. It argues that there are, mostly, three different types of people within the Semantic Web for the Life Sciences arena (it may just as well be applicable to another subject domain if they were to experiment with SWT, e.g., in public administration): the AI researchers, the philosophers, and the IT-savvy domain experts. They each have their own motivations and goals, which, at times, clash, but with conversation, respect, understanding, compromise, and collaboration, one will, and can, achieve the realisation of theory and ideas in useful applications.

[10] `http://www.sofg.org/meetings/sofg2004/Goble.ppt`

[11] `http://www.sofg.org/`

Bibliography

[Acciarri et al., 2005] Acciarri, A., Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Palmieri, M., and Rosati, R. (2005). QuOnto: Querying Ontologies. In *Proc. of the 20th Nat. Conf. on Artificial Intelligence (AAAI 2005)*, pages 1670–1671.

[Alberts et al., 2008] Alberts, R., Calvanese, D., Giacomo, G. D., Gerber, A., Horridge, M., Kaplunova, A., Keet, C. M., Lembo, D., Lenzerini, M., Milicic, M., Möller, R., Rodríguez-Muro, M., Rosati, R., Sattler, U., Suntisrivaraporn, B., Stefanoni, G., Turhan, A.-Y., Wandelt, S., and Wessel, M. (2008). Analysis of test results on usage scenarios. Deliverable TONES-D27 v1.0, TONES Project.

[Alberts et al., 2012] Alberts, R., Fogwill, T., and Keet, C. M. (2012). Several required OWL features for indigenous knowledge management systems. In Klinov, P. and Horridge, M., editors, *7th Workshop on OWL: Experiences and Directions (OWLED 2012)*, volume 849 of *CEUR-WS*, page 12p. 27-28 May, Heraklion, Crete, Greece.

[Alexopoulou et al., 2008] Alexopoulou, D., Wächter, T., Pickersgill, L., Eyre, C., and Schroeder, M. (2008). Terminologies for text-mining; an experiment in the lipoprotein metabolism domain. *BMC Bioinformatics*, 9(Suppl 4):S2.

[Aliseda, 2004] Aliseda, A. (2004). Logics in scientific discovery. *Foundation of Science*, 9:339–363.

[Allen, 1983] Allen, J. F. (1983). Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843.

[Antoniou and van Harmelen, 2003] Antoniou, G. and van Harmelen, F. (2003). *A Semantic Web Primer*. MIT Press.

[Artale et al., 2007a] Artale, A., Calvanese, D., Kontchakov, R., Ryzhikov, V., and Zakharyaschev, M. (2007a). Reasoning over extended ER models. In Parent, C., Schewe, K.-D., Storey, V. C., and Thalheim, B., editors, *Proceedings of the 26th International Conference on Conceptual Modeling (ER'07)*, volume 4801 of *LNCS*, pages 277–292. Springer. Auckland, New Zealand, November 5-9, 2007.

[Artale et al., 2009] Artale, A., Calvanese, D., Kontchakov, R., and Zakharyaschev, M. (2009). Dl-lite without the unique name assumption. In *Proc. of the 22nd Int. Workshop on Description Logic (DL 2009)*, volume 477 of *CEUR-WS*. http://ceur-ws.org/.

[Artale et al., 2002] Artale, A., Franconi, E., Wolter, F., and Zakharyaschev, M. (2002). A temporal description logic for reasoning about conceptual schemas and queries. In Flesca, S., Greco, S., Leone, N., and Ianni, G., editors, *Proceedings of the 8th Joint European Conference on Logics in Artificial Intelligence (JELIA-02)*, volume 2424 of *LNAI*, pages 98–110. Springer Verlag.

[Artale et al., 2008] Artale, A., Guarino, N., and Keet, C. M. (2008). Formalising temporal constraints on part-whole relations. In Brewka, G. and Lang, J., editors, *11th International Conference on Principles of Knowledge Representation and Reasoning (KR'08)*, pages 673–683. AAAI Press. Sydney, Australia, September 16-19, 2008.

[Artale et al., 2007b] Artale, A., Kontchakov, R., Lutz, C., Wolter, F., and Zakharyaschev, M. (2007b). Temporalising tractable description logics. In *Inter. Symposium on Temporal Representation and Reasoning (TIME07)*. IEEE Computer Society.

[Artale et al., 2007c] Artale, A., Parent, C., and Spaccapietra, S. (2007c). Evolving objects in temporal information systems. *Annals of Mathematics and Artificial Intelligence*, 50(1-2):5–38.

[Baader et al., 2005] Baader, F., Brandt, S., and Lutz., C. (2005). Pushing the EL envelope. In *Proc. of the 19th Joint Int. Conf. on Artificial Intelligence (IJCAI 2005)*.

[Baader et al., 2008] Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D., and Patel-Schneider, P. F., editors (2008). *The Description Logics Handbook – Theory and Applications*. Cambridge University Press, 2 edition.

[Baader et al., 2007a] Baader, F., Ganter, B., Sertkaya, B., and Sattler, U. (2007a). Completing description logic knowledge bases using formal concept analysis. In *Proc. of IJCAI 2007*. Hyderabad, India, 2007.

[Baader et al., 2007b] Baader, F., Sertkaya, B., and Turhan, A.-Y. (2007b). Computing the least common subsumer w.r.t. a background terminology. *Journal of Applied Logic*, 5(3):392–420.

[Barwise and Etchemendy, 1993] Barwise, J. and Etchemendy, J. (1993). *The language of first-order logic*. Stanford, USA: CSLI Lecture Notes, 3rd edition.

[Berardi et al., 2005] Berardi, D., Calvanese, D., and De Giacomo, G. (2005). Reasoning on UML class diagrams. *Artificial Intelligence*, 168(1-2):70–118.

[Berners-Lee et al., 2001] Berners-Lee, T., Hendler, J., and Lassila, O. (2001). The semantic web. *Scientific American Magazine*, May 17, 2001.

[Biasiotti and Fernández-Barrera, 2009] Biasiotti, M. A. and Fernández-Barrera, M. (2009). Enriching thesauri with ontological information: Eurovoc thesaurus and DALOS domain ontology of consumer law. In *Proceedings of the Third Workshop on Legal Ontologies and Artificial Intelligence Techniques (LOAIT 2009)*. Barcelona, Spain, June 8, 2009.

[Bloesch and Halpin, 1996] Bloesch, A. C. and Halpin, T. A. (1996). ConQuer: a conceptual query language. In *Proceedings of ER'96: 15th International Conference onconceptual modeling*, volume 1157 of *LNCS*, pages 121–133. Springer.

[Bloesch and Halpin, 1997] Bloesch, A. C. and Halpin, T. A. (1997). Conceptual Queries using ConQuer-II. In *Proceedings of ER'97: 16th International Conference on Conceptual Modeling*, volume 1331 of *LNCS*, pages 113–126. Springer.

[Bobillo and Straccia, 2011] Bobillo, F. and Straccia, U. (2011). Fuzzy ontology representation using OWL 2. *International Journal of Approximate Reasoning*, 52:1073–1094.

[Borgida et al., 2008] Borgida, A., Calvanese, D., and Rodríguez-Muro, M. (2008). Explanation in the DL-Lite family of description logics. In *Proc. of the 7th Int. Conf. on Ontologies, DataBases, and Applications of Semantics (ODBASE 2008)*, volume 5332 of *LNCS*, pages 1440–1457. Springer.

[Borgo and Masolo, 2009] Borgo, S. and Masolo, C. (2009). Foundational choices in DOLCE. In Staab, S. and Studer, R., editors, *Handbook on Ontologies*, pages 361–381. Springer, 2 edition.

[Bornberg-Bauer and Paton, 2002] Bornberg-Bauer, E. and Paton, N. (2002). Conceptual data modelling for bioinformatics. *Briefings in Bioinformatics*, 3(2):166180.

[Bouayad-Agha et al., 2014] Bouayad-Agha, N., Casamayor, G., and Wanner, L. (2014). Natural language generation in the context of the semantic web. *Semantic Web Journal*, 5(6):493–513.

[Brochhausen, 2006] Brochhausen, M. (2006). The *Derives_from* relation in biomedical ontologies. *Studies in Health Technology and Informatics*, 124:769–774.

[Buitelaar et al., 2009] Buitelaar, P., Cimiano, P., Haase, P., and M., S. (2009). Towards linguistically grounded ontologies. In *Proceedings of the Extended Semantic Web Conference (ESWC;09)*, LNCS. Springer.

[Calbimonte et al., 2010] Calbimonte, J.-P., Corcho, O., and Gray, A. J. G. (2010). Enabling ontology-based access to streaming data sources. In *9th International Semantic Web Conference (ISWC'10)*, volume 6496 of *LNCS*. Springer. Shanghai, China, Nov 7-11.

[Calvanese and De Giacomo, 2003] Calvanese, D. and De Giacomo, G. (2003). *The DL Handbook: Theory, Implementation and Applications*, chapter Expressive description logics, pages 178–218. Cambridge University Press.

[Calvanese et al., 1998a] Calvanese, D., De Giacomo, G., Lenzerini, M., Nardi, D., and Rosati, R. (1998a). Description logic framework for information integration. In *Proc. of the 6th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR98)*, pages 2–13.

[Calvanese et al., 2009] Calvanese, D., Giacomo, G. D., Lembo, D., Lenzerini, M., Poggi, A., Rodríguez-Muro, M., and Rosati, R. (2009). Ontologies and databases: The DL-Lite approach. In Tessaris, S. and Franconi, E., editors, *Semantic Technologies for Informations Systems - 5th Int. Reasoning Web Summer School (RW 2009)*, volume 5689 of *LNCS*, pages 255–356. Springer. Brixen-Bressanone, Italy, 30 August - 4 September 2009.

[Calvanese et al., 2007] Calvanese, D., Giacomo, G. D., Lembo, D., Lenzerini, M., and Rosati, R. (2007). Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *Journal of Automated Reasoning*, 39(3):385–429.

[Calvanese et al., 2010] Calvanese, D., Keet, C. M., Nutt, W., Rodríguez-Muro, M., and Stefanoni, G. (2010). Web-based graphical querying of databases through an ontology: the WONDER system. In Shin, S. Y., Ossowski, S., Schumacher, M., Palakal, M. J., and Hung, C.-C., editors, *Proceedings of ACM Symposium on Applied Computing (ACM SAC'10)*, pages 1389–1396. ACM. March 22-26 2010, Sierre, Switzerland.

[Calvanese et al., 1998b] Calvanese, D., Lenzerini, M., and Nardi, D. (1998b). *Logics for Databases and Information Systems*, chapter Description logics for conceptual data modeling. Kluwer, Amsterdam.

[Calvanese et al., 1999] Calvanese, D., Lenzerini, M., and Nardi, D. (1999). Unifying class-based representation formalisms. *Journal of Artificial Intelligence Research*, 11:199–240.

[Catarci and Santucci, 1994] Catarci, T. and Santucci, G. (1994). Query by diagram: a graphical environment for querying databases. *ACM SIGMOD Record*, 23(2):515.

[Chavula and Keet, 2014] Chavula, C. and Keet, C. M. (2014). Is lemon sufficient for building multilingual ontologies for Bantu languages? In Keet, C. M. and Tamma, V., editors, *Proceedings of the 11th OWL: Experiences and Directions Workshop (OWLED'14)*, volume 1265 of *CEUR-WS*, pages 61–72. Riva del Garda, Italy, Oct 17-18, 2014.

[Chomicki and Toman, 1998] Chomicki, J. and Toman, D. (1998). *Logics for databases and information systems*, chapter Temporal logic in information systems. Kluwer.

[Corcho et al., 2003] Corcho, O., Fernandez-Lopez, M., and Gómez-Pérez, A. (2003). Methodologies, tools and languages for building ontologies. where is their meeting point? *Data & Knowledge Engineering*, 46(1):41–64.

[Corcho and Mariano Fernández-López, 2005] Corcho, O. and Mariano Fernández-López, Asunción Gómez-Pérez, A. L.-C. (2005). Building legal ontologies with methontology and webode. In *Law and the Semantic Web 2005*, volume 3369 of *LNAI*, pages 142–157. Springer LNAI.

[Coulet et al., 2010] Coulet, A., Shah, N. H., Garten, Y., Musen, M., and Altman, R. B. (2010). Using text to build semantic networks for pharmacogenomics. *Journal of Biomedical Informatics*, 43(6):1009–1019.

[Cuenca Grau et al., 2008] Cuenca Grau, B., Horrocks, I., Motik, B., Parsia, B., Patel-Schneider, P., and Sattler, U. (2008). OWL 2: The next step for OWL. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, 6(4):309–322.

[da Costa and Laskey, 2006] da Costa, P. C. G. and Laskey, K. B. (2006). PR-OWL: A framework for probabilistic ontologies. In *Proceedings FOIS'06*, pages 237–249. IOS Press.

[d'Amato et al., 201x] d'Amato, C., Fanizzi, N., and Esposito, F. (201x). Inductive learning for the Semantic Web: What does it buy? *Semantic Web Journal*, page accepted. latest version: http://www.semantic-web-journal.net/content/inductive-learning-semantic-web-what-does-it-buy.

[Demir et al., 2010] Demir, E. et al. (2010). The BioPAX community standard for pathway data sharing. *Nature Biotechnology*, 28(9):935–942.

[Dietze et al., 2008] Dietze, H., Alexopoulou, D., Alvers, M. R., Barrio-Alvers, L., Andreopoulos, B., Doms, A., Hakenberg, J., Moennich, J., Plake, C., Reischuck, A., Royer, L., Waechter, T., Zschunke, M., and Schroeder, M. (2008). Gopubmed: Exploring pubmed with ontological background knowledge. In Krawetz, S. A., editor, *Bioinformatics for Systems Biology*. Humana Press.

[El-Ghalayini et al., 2006] El-Ghalayini, H., Odeh, M., McClatchey, R., and Arnold, D. (2006). Deriving conceptual data models from domain ontologies for bioinformatics. In *2nd Conference on Information and Communication Technologies (ICTTA'06)*, pages 3562 – 3567. IEEE Computer Society. 24-28 April 2006, Damascus, Syria.

[Euzenat and Montanari, 2005] Euzenat, J. and Montanari, A. (2005). *Handbook of temporal reasoning in artificial intelligence*, chapter Time granularity, pages 59–118. Amsterdam: Elsevier.

[Euzenat and Shvaiko, 2007] Euzenat, J. and Shvaiko, P. (2007). *Ontology Matching*. Springer.

[Fernández et al., 1999] Fernández, M., Gómez-Pérez, A., Pazos, A., and Pazos, J. (1999). Building a chemical ontology using METHONTOLOGY and the ontology design environment. *IEEE Expert: Special Issue on Uses of Ontologies*, January/February:37–46.

[Ferré and Rudolph, 2012] Ferré, S. and Rudolph, S. (2012). Advocatus diaboli exploratory enrichment of ontologies with negative constraints. In ten Teije, A. et al., editors, *18th International Conference on Knowledge Engineering and Knowledge Management (EKAW'12)*, volume 7603 of *LNAI*, pages 42–56. Springer. Oct 8-12, Galway, Ireland.

[Fillottrani et al., 2012] Fillottrani, P. R., Franconi, E., and Tessaris, S. (2012). The ICOM 3.0 intelligent conceptual modelling tool and methodology. *Semantic Web Journal*, 3(3):293–306.

[Franscescomarino et al., 2012] Franscescomarino, C. D., Ghidini, C., and Rospocher, M. (2012). Evaluating wiki-enhanced ontology authoring. In ten Teije, A. et al., editors, *18th International Conference on Knowledge Engineering and Knowledge Management (EKAW'12)*, volume 7603 of *LNAI*, pages 292–301. Springer. Oct 8-12, Galway, Ireland.

[Fuchs et al., 2010] Fuchs, N. E., Kaljurand, K., and Kuhn, T. (2010). Discourse Representation Structures for ACE 6.6. Technical Report ifi-2010.0010, Department of Informatics, University of Zurich, Zurich, Switzerland.

[Gabbay et al., 2003] Gabbay, D., Kurucz, A., Wolter, F., and Zakharyaschev, M. (2003). *Many-dimensional modal logics: theory and applications*. Studies in Logic. Elsevier.

[Garcia et al., 2010] Garcia, A., O'Neill, K., Garcia, L. J., Lord, P., Stevens, R., Corcho, O., and Gibson, F. (2010). Developing ontologies within decentralized settings. In Chen, H. et al., editors, *Semantic e-Science. Annals of Information Systems 11*, pages 99–139. Springer.

[Gene Ontology Consortium, 2000] Gene Ontology Consortium (2000). Gene Ontology: tool for the unification of biology. *Nature Genetics*, 25:25–29.

[Gennari et al., 2003] Gennari, J. H., Musen, M. A., Fergerson, R. W., Grosso, W. E., Crubézy, M., Eriksson, H., Noy, N. F., and Tu, S. W. (2003). The evolution of Protégé: an environment for knowledge-based systems development. *International Journal of Human-Computer Studies*, 58(1):89–123.

[Ghidini et al., 2009] Ghidini, C., Kump, B., Lindstaedt, S., Mabhub, N., Pammer, V., Rospocher, M., and Serafini, L. (2009). Moki: The enterprise modelling wiki. In *Proceedings of the 6th Annual European Semantic Web Conference (ESWC2009).* Heraklion, Greece, 2009 (demo).

[Glimm et al., 2010] Glimm, B., Rudolph, S., and Völker, J. (2010). Integrated metamodeling and diagnosis in OWL 2. In Patel-Schneider, P. F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J. Z., Horrocks, I., and Glimm, B., editors, *Proceedings of the 9th International Semantic Web Conference*, volume 6496 of *LNCS*, pages 257–272. Springer.

[Goble et al., 2007] Goble, C., Wolstencroft, K., Goderis, A., Hull, D., Zhao, J., Alper, P., Lord, P., Wroe, C., Belhajjame, K., Turi, D., Stevens, R., Oinn, T., and Roure, D. D. (2007). Knowledge discovery for biology with taverna. In Baker, C. and Cheung, H., editors, *Semantic Web: Revolutionizing knowledge discovery in the life sciences*, pages 355–395. Springer: New York.

[Goble and Wroe, 2004] Goble, C. and Wroe, C. (2004). The montagues and the capulets. *Comparative and Functional Genomics*, 5(8):623–632.

[Golbreich and Horrocks, 2007] Golbreich, C. and Horrocks, I. (2007). The OBO to OWL mapping, GO to OWL 1.1! In *Proc. of the Third OWL Experiences and Directions Workshop*, volume 258 of *CEUR-WS*. http://ceur-ws.org/.

[Gómez-Pérez et al., 2004] Gómez-Pérez, A., Fernández-Lopez, M., and Corcho, O. (2004). *Ontological Engineering.* Springer Verlag.

[Grosof et al., 2003] Grosof, B. N., Horrocks, I., Volz, R., and Decker, S. (2003). Description logic programs: Combining logic programs with description logic. In *Proc. of the 12th Int. World Wide Web Conference (WWW'03)*, pages 48–57. Budapest, Hungary.

[Gruber, 1993] Gruber, T. R. (1993). A translation approach to portable ontologies. *Knowledge Acquisition*, 5(2):199–220.

[Grütter and Bauer-Messmer, 2007] Grütter, R. and Bauer-Messmer, B. (2007). Combining OWL with RCC for spatioterminological reasoning on environmental data. In *Third international Workshop OWL: Experiences and Directions (OWLED 2007).* 6-7 June 2007, Innsbruck, Austria.

[Guarino, 1998] Guarino, N. (1998). Formal ontology and information systems. In Guarino, N., editor, *Proceedings of Formal Ontology in Information Systems (FOIS'98)*, Frontiers in Artificial intelligence and Applications, pages 3–15. Amsterdam: IOS Press.

[Guarino, 2009] Guarino, N. (2009). The ontological level: Revisiting 30 years of knowledge representation. In Borgida, A. et al., editors, *Mylopoulos Festschrift*, volume 5600 of *LNCS*, pages 52–67. Springer.

[Guarino et al., 2009] Guarino, N., Oberle, D., and Staab, S. (2009). What is an ontology? In Staab, S. and Studer, R., editors, *Handbook on Ontologies*, chapter 1, pages 1–17. Springer.

[Guarino and Welty, 2000a] Guarino, N. and Welty, C. (2000a). A formal ontology of properties. In Dieng, R. and Corby, O., editors, *Proceedings of 12th International Conference on Knowledge Engineering and Knowledge Management (EKAW'00)*, volume 1937 of *LNCS*, pages 97–112. Springer Verlag.

[Guarino and Welty, 2000b] Guarino, N. and Welty, C. (2000b). Identity, unity, and individuality: towards a formal toolkit for ontological analysis. In Horn, W., editor, *Proceedings of ECAI'00*, pages 219–223. IOS Press, Amsterdam.

[Guarino and Welty, 2009] Guarino, N. and Welty, C. (2009). An overview of ontoclean. In Staab, S. and Studer, R., editors, *Handbook on Ontologies*, pages 201–220. Springer Verlag.

[Guizzardi, 2005] Guizzardi, G. (2005). *Ontological Foundations for Structural Conceptual Models.* Phd thesis, University of Twente, The Netherlands. Telematica Instituut Fundamental Research Series No. 15.

[Hainaut et al., 1993] Hainaut, J.-L., Chandelon, M., Tonneau, C., and Joris, M. (1993). Contribution to a theory of database reverse engineering. In *Reverse Engineering, 1993., Proceedings of Working Conference on*, pages 161–170.

[Halpin, 2001] Halpin, T. (2001). *Information Modeling and Relational Databases.* San Francisco: Morgan Kaufmann Publishers.

[Hedman, 2004] Hedman, S. (2004). *A first course in logic—an introduction to model theory, proof theory, computability, and complexity.* Oxford University Press, Oxford.

[Henze et al., 2004] Henze, N., Dolog, P., and Nejdl, W. (2004). Reasoning and ontologies for personalized e-learning in the semantic web. *Educational Technology & Society*, 7(4):82–97.

[Hepp, 2011] Hepp, M. (Last accessed: Aug 30, 2011). SKOS to OWL. Online.

[Herre and Heller, 2006] Herre, H. and Heller, B. (2006). Semantic foundations of medical information systems based on top-level ontologies. *Knowledge-Based Systems*, 19:107–115.

[Hilario et al., 2011] Hilario, M., Nguyen, P., Do, H., Woznica, A., and Kalous, A. (2011). Ontology-based meta-mining of knowledge discovery workflows. In Jankowski, N., Duch, W., and Grabczewski, K., editors, *Meta-learning in Computational Intelligence*, pages 273–315. Springer.

[Hobbs and Pan, 2004] Hobbs, J. R. and Pan, F. (2004). An ontology of time for the semantic web. *ACM Transactions on Asian Language Processing (TALIP): Special issue on Temporal Information Processing*, 3(1):66–85.

[Hodgkinson et al., 1999] Hodgkinson, I. M., Wolter, F., and Zakharyaschev, M. (1999). Decidable fragments of first-order temporal logics. *Annals of pure and applied logic*, 106:85–134.

[Hoehndorf et al., 2011] Hoehndorf, R., Dumontier, M., Gennari, J. H., Wimalaratne, S., de Bono, B., Cook, D., and Gkoutos, G. (2011). Integrating systems biology models and biomedical ontologies. *BMC Systems Biology*, 5:124.

[Hoehndorf et al., 2010] Hoehndorf, R., Oellrich, A., Dumontier, M., Kelso, J., Rebholz-Schuhmann, D., and Herre, H. (2010). Relations as patterns: bridging the gap between OBO and OWL. *BMC Bioinformatics*, 11(1):441.

[Hofstede and Proper, 1998] Hofstede, A. H. M. t. and Proper, H. A. (1998). How to formalize it? formalization principles for information systems development methods. *Information and Software Technology*, 40(10):519–540.

[Horridge et al., 2008] Horridge, M., Parsia, B., and Sattler, U. (2008). Laconic and precise justifications in OWL. In *Proc. of the 7th International Semantic Web Conference (ISWC 2008)*, volume 5318 of *LNCS*. Springer.

[Horrocks et al., 2006] Horrocks, I., Kutz, O., and Sattler, U. (2006). The even more irresistible $\mathcal{SROIQ}$. *Proceedings of KR-2006*, pages 452–457.

[Horrocks et al., 2003] Horrocks, I., Patel-Schneider, P. F., and van Harmelen, F. (2003). From SHIQ and RDF to OWL: The making of a web ontology language. *Journal of Web Semantics*, 1(1):7.

[Isaac and Summers, 2009] Isaac, A. and Summers, E. (2009). SKOS Simple Knowledge Organization System Primer. W3c standard, World Wide Web Consortium. http://www.w3.org/TR/skos-primer.

[Jarrar et al., 2003] Jarrar, M., Demy, J., and Meersman, R. (2003). On using conceptual data modeling for ontology engineering. *Journal on Data Semantics: Special issue on Best papers from the ER/ODBASE/COOPIS 2002 Conferences*, 1(1):185–207.

[Jarrar et al., 2006] Jarrar, M., Keet, C. M., and Dongilli, P. (2006). Multilingual verbalization of ORM conceptual models and axiomatized ontologies. Starlab technical report, Vrije Universiteit Brussel, Belgium.

[Jiang et al., 2009] Jiang, Y., Wang, J., Tang, S., and Xiao, B. (2009). Reasoning with rough description logics: An approximate concepts approach. *Information Sciences*, 179:600–612.

[Kassel, 2005] Kassel, G. (2005). Integration of the DOLCE top-level ontology into the OntoSpec methodology. Technical Report HAL : hal-00012203/arXiv : cs.AI/0510050, Laboratoire de Recherche en Informatique d'Amiens (LaRIA). http://hal.archives-ouvertes.fr/ccsd-00012203.

[Kazakov, 2008] Kazakov, Y. (2008). RIQ and SROIQ are harder than SHOIQ. In *11th International Conference on Principles of Knowledge Representation and Reasoning (KR'08)*, pages 274–284. 16-19 August 2008, Sydney, Australia.

[Keet and Khumalo, 2014a] Keet, C. and Khumalo, L. (2014a). Basics for a grammar engine to verbalize logical theories in isiZulu. In Bikakis, A. et al., editors, *Proceedings of the 8th International Web Rule Symposium (RuleML'14)*, volume 8620 of *LNCS*, pages 216–225. Springer. August 18-20, 2014, Prague, Czech Republic.

[Keet and Khumalo, 2014b] Keet, C. and Khumalo, L. (2014b). Toward verbalizing logical theories in isiZulu. In Davis, B., Kuhn, T., and Kaljurand, K., editors, *Proceedings of the 4th Workshop on Controlled Natural Language (CNL'14)*, volume 8625 of *LNAI*, pages 78–89. Springer. 20-22 August 2014, Galway, Ireland.

[Keet et al., 2015] Keet, C., Lawrynowicz, A., d'Amato, C., Kalousis, A., Nguyen, P., Palma, R., Stevens, R., and Hilario, M. (2015). The data mining optimization ontology. *Web Semantics: Science, Services and Agents on the World Wide Web*, page in print.

[Keet, 2005] Keet, C. M. (2005). Factors affecting ontology development in ecology. In Ludäscher, B. and Raschid, L., editors, *Data Integration in the Life Sciences 2005 (DILS2005)*, volume 3615 of *LNBI*, pages 46–62. Springer Verlag. San Diego, USA, 20-22 July 2005.

[Keet, 2008] Keet, C. M. (2008). A formal comparison of conceptual data modeling languages. In *13th International Workshop on Exploring Modeling Methods in Systems Analysis and Design (EMMSAD'08)*, volume 337 of *CEUR-WS*, pages 25–39. 16-17 June 2008, Montpellier, France.

[Keet, 2009a] Keet, C. M. (2009a). Constraints for representing transforming entities in bio-ontologies. In Serra, R. and Cucchiara, R., editors, *11th Congress of the Italian Association for Artificial Intelligence (AI*IA 2009)*, volume 5883 of *LNAI*, pages 11–20. Springer Verlag. Reggio Emilia, Italy, Dec. 9-12, 2009.

[Keet, 2009b] Keet, C. M. (2009b). Mapping the Object-Role Modeling language ORM2 into Description Logic language $\mathcal{DLR}_{ifd}$. Technical Report arXiv:cs.LO/0702089v2, KRDB Research Centre, Free University of Bozen-Bolzano, Italy. arXiv:cs.LO/0702089v2.

[Keet, 2009c] Keet, C. M. (2009c). Ontology design parameters for aligning agri-informatics with the semantic web. In Sartori, F., Sicilia, M., and Manouselis, N., editors, *3rd International Conference on Metadata and Semantics (MTSR'09) – Special Track on Agriculture, Food & Environment*, volume 46 of *CCIS*, pages 239–244. Springer. Oct 1-2 2009 Milan, Italy.

[Keet, 2010a] Keet, C. M. (2010a). Dependencies between ontology design parameters. *International Journal of Metadata, Semantics and Ontologies*, 5(4):265–284.

[Keet, 2010b] Keet, C. M. (2010b). On the feasibility of description logic knowledge bases with rough concepts and vague instances. In *Proceedings of the 23rd International Workshop on Description Logics (DL'10)*, CEUR-WS, pages 314–324. 4-7 May 2010, Waterloo, Canada.

[Keet, 2010c] Keet, C. M. (2010c). Ontology engineering with rough concepts and instances. In Cimiano, P. and Pinto, H., editors, *17th International Conference on Knowledge Engineering and Knowledge Management (EKAW'10)*, volume 6317 of *LNCS*, pages 507–517. Springer. 11-15 October 2010, Lisbon, Portugal.

[Keet, 2011a] Keet, C. M. (2011a). Rough subsumption reasoning with rOWL. In *Proceeding of the SAICSIT Annual Research Conference 2011 (SAICSIT'11)*, pages 133–140. ACM Conference Proceedings. Cape Town, South Africa, October 3-5, 2011.

[Keet, 2011b] Keet, C. M. (2011b). The use of foundational ontologies in ontology development: an empirical assessment. In Antoniou, G. et al., editors, *8th Extended Semantic Web Conference (ESWC'11)*, volume 6643 of *LNCS*, pages 321–335. Springer. Heraklion, Crete, Greece, 29 May-2 June, 2011.

[Keet, 2012a] Keet, C. M. (2012a). Detecting and revising flaws in OWL object property expressions. In ten Teije, A. et al., editors, *18th International Conference on Knowledge Engineering and Knowledge Management (EKAW'12)*, volume 7603 of *LNAI*, pages 252–266. Springer. Oct 8-12, Galway, Ireland.

[Keet, 2012b] Keet, C. M. (2012b). Transforming semi-structured life science diagrams into meaningful domain ontologies with DiDOn. *Journal of Biomedical Informatics*, 45:482–494.

[Keet, 2013] Keet, C. M. (2013). Ontology-driven formal conceptual data modeling for biological data analysis. In Elloumi, M. and Zomaya, A. Y., editors, *Biological Knowledge Discovery Handbook: Preprocessing, Mining and Postprocessing of Biological Data*, chapter 6, pages 129–154. Wiley.

[Keet and Artale, 2008] Keet, C. M. and Artale, A. (2008). Representing and reasoning over a taxonomy of part-whole relations. *Applied Ontology – Special issue on Ontological Foundations for Conceptual Modeling*, 3(1-2):91–110.

[Keet and Artale, 2010] Keet, C. M. and Artale, A. (2010). A basic characterization of relation migration. In Meersman, R. et al., editors, *OTM Workshops, 6th International Workshop on Fact-Oriented Modeling (ORM'10)*, volume 6428 of *LNCS*, pages 484–493. Springer. October 27-29, 2010, Hersonissou, Crete, Greece.

[Keet et al., 2012] Keet, C. M., Fernández-Reyes, F. C., and Morales-González, A. (2012). Representing mereotopological relations in OWL ontologies with ONTOPARTS. In Simperl, E. et al., editors, *Proceedings of the 9th Extended Semantic Web Conference (ESWC'12)*, volume 7295 of *LNCS*, pages 240–254. Springer. 29-31 May 2012, Heraklion, Crete, Greece.

[Keet et al., 2013a] Keet, C. M., Khan, M. T., and Ghidini, C. (2013a). Ontology authoring with FORZA. In *Proceedings of the 22nd ACM international conference on Conference on Information & Knowledge Management (CIKM'13)*, pages 569–578. ACM proceedings. Oct. 27 - Nov. 1, 2013, San Francisco, USA.

[Keet et al., 2013b] Keet, C. M., Lawrynowicz, A., d'Amato, C., and Hilario, M. (2013b). Modeling issues and choices in the Data Mining OPtimisation Ontology. In *8th Workshop on OWL: Experiences and Directions (OWLED'13)*, volume 1080 of *CEUR-WS*. 26-27 May 2013, Montpellier, France.

[Keet and Rodríguez, 2007] Keet, C. M. and Rodríguez, M. (2007). Toward using biomedical ontologies: trade-offs between ontology languages. In *AAAI 2007 Workshop Semantic eScience (SeS 2007)*, volume WS-07-11 of *AAAI Technical Report*, pages 65–68. AAAI. 23 July 2007, Vancouver, Canada.

[Keet et al., 2007] Keet, C. M., Roos, M., and Marshall, M. S. (2007). A survey of requirements for automated reasoning services for bio-ontologies in OWL. In *Proceedings of the 3rd Workshop on OWL: Experiences and Directions (OWLED 2007)*, volume 258 of *CEUR-WS*. 6-7 June 2007, Innsbruck, Austria.

[Keet et al., 2013c] Keet, C. M., Suárez-Figueroa, M. C., and Poveda-Villalón, M. (2013c). The current landscape of pitfalls in ontologies. In Filipe, J. and Dietz, J., editors, *5th International Conference on Knowledge Engineering and Ontology Development (KEOD'13)*, pages 132–139. INSTICC, SCITEPRESS – Science and Technology Publications. Vilamoura, Portugal, 19-22 September 2013.

[Khan and Keet, 2012] Khan, Z. and Keet, C. M. (2012). ONSET: Automated foundational ontology selection and explanation. In ten Teije, A. et al., editors, *18th International Conference on Knowledge Engineering and Knowledge Management (EKAW'12)*, volume 7603 of *LNAI*, pages 237–251. Springer. Oct 8-12, Galway, Ireland.

[Khan and Keet, 2013a] Khan, Z. and Keet, C. M. (2013a). Addressing issues in foundational ontology mediation. In Filipe, J. and Dietz, J., editors, *5th International Conference on Knowledge Engineering and Ontology Development (KEOD'13)*, pages 5–16. INSTICC, SCITEPRESS – Science and Technology Publications. Vilamoura, Portugal, 19-22 September 2013.

[Khan and Keet, 2013b] Khan, Z. and Keet, C. M. (2013b). The foundational ontology library romulus. In Cuzzocrea, A. and Maabout, S., editors, *Proceedings of the 3rd International Conference on Model & Data Engineering (MEDI'13)*, volume 8216 of *LNCS*, pages 200–211. Springer. September 25-27, 2013, Amantea, Calabria, Italy.

[Khan and Keet, 2013c] Khan, Z. and Keet, C. M. (2013c). Toward semantic interoperability with aligned foundational ontologies in ROMULUS. In Benjamins, R., d'Aquin, M., Gordon, A., , and Gómez-Pérez, J. M., editors, *Seventh International Conference on Knowledge Capture (K-CAP'13)*, page a27 (poster&demo). ACM. 23-26 June 2013, Banff, Canada.

[Kontchakov et al., 2010] Kontchakov, R., Lutz, C., Toman, D., Wolter, F., and Zakharyaschev, M. (2010). The combined approach to query answering in DL-Lite. In Lin, F., Sattler, U., and Truszczynski, M., editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Twelfth International Conference (KR 2010)*. AAAI Press. Toronto, Ontario, Canada, May 9-13, 2010.

[Krötzsch et al., 2012] Krötzsch, M., Simančík, F., and Horrocks, I. (2012). A description logic primer. Technical Report 1201.4089v1, Department of Computer Science, University of Oxford, UK. arXiv:cs.LO/12014089v1.

[Liu et al., 2011] Liu, K., Hogan, W. R., and Crowley, R. S. (2011). Natural language processing methods and systems for biomedical ontology learning. *Journal of Biomedical Informatics*, 44(1):163–179.

[Lubyte and Tessaris, 2009] Lubyte, L. and Tessaris, S. (2009). Automated extraction of ontologies wrapping relational data sources. In *Proc. of DEXA'09*.

[Lukasiewicz and Straccia, 2008] Lukasiewicz, T. and Straccia, U. (2008). Managing uncertainty and vagueness in description logics for the semantic web. *Journal of Web Semantics*, 6(4):291–308.

[Lutz et al., 2009] Lutz, C., Toman, D., and Wolter, F. (2009). Conjunctive query answering in the description logic EL using a relational database system. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence IJCAI'09*. AAAI Press.

[Lutz et al., 2008] Lutz, C., Wolter, F., and Zakharyaschev, M. (2008). Temporal description logics: A survey. In *Proc. of the Fifteenth International Symposium on Temporal Representation and Reasoning (TIME'08)*. IEEE Computer Society Press.

[MacLeod and Rubenstein, 2005] MacLeod, M. C. and Rubenstein, E. M. (2005). Universals. In *The Internet Encyclopedia of Philosophy*. http://www.iep.utm.edu/u/universa.htm.

[Madin et al., 2008] Madin, J. S., Bowers, S., Schildhauer, M. P., and Jones, M. B. (2008). Advancing ecological research with ontologies. *Trends in Ecology & Evolution*, 23(3):159–168.

[Masolo et al., 2003] Masolo, C., Borgo, S., Gangemi, A., Guarino, N., and Oltramari, A. (2003). Ontology library. WonderWeb Deliverable D18 (ver. 1.0, 31-12-2003). http://wonderweb.semanticweb.org.

[McCrae et al., 2012] McCrae, J., de Cea, G. A., Buitelaar, P., Cimiano, P., Declerck, T., Gómez-Pérez, A., Gracia, J., Hollink, L., Montiel-Ponsoda, E., Spohr, D., and Wunner, T. (2012). The lemon cookbook. Technical report, Monnet Project. www.lemon-model.net.

[Merrill, 2010a] Merrill, G. H. (2010a). Ontological realism: Methodology or misdirection? *Applied Ontology*, 5(2):79108.

[Merrill, 2010b] Merrill, G. H. (2010b). Realism and reference ontologies: Considerations, reflections and problems. *Applied Ontology*, 5(3):189–221.

[Miles and Bechhofer, 2009] Miles, A. and Bechhofer, S. (2009). SKOS Simple Knowledge Organization System Reference. W3c recommendation, World Wide Web Consortium (W3C).

[Motik et al., 2009a] Motik, B., Grau, B. C., Horrocks, I., Wu, Z., Fokoue, A., and Lutz, C. (2009a). OWL 2 Web Ontology Language Profiles. W3C recommendation, W3C. http://www.w3.org/TR/owl2-profiles/.

[Motik et al., 2009b] Motik, B., Patel-Schneider, P. F., and Parsia, B. (2009b). OWL 2 web ontology language structural specification and functional-style syntax. W3c recommendation, W3C. http://www.w3.org/TR/owl2-syntax/.

[Nikitin et al., 2003] Nikitin, A., Egorov, S., Daraselia, N., and Mazo, I. (2003). Pathway studio–the analysis and navigation of molecular networks. *Bioinformatics*, 19(16):2155–2157.

[Noy and McGuinness, 2001] Noy, N. and McGuinness, D. (2001). Ontology development 101: A guide to creating your first ontology. Technical Report KSL-01-05, and Stanford Medical Informatics Technical Report SMI-2001-0880, Stanford Knowledge Systems Laboratory.

[Parsia et al., 2005] Parsia, B., Sirin, E., and Kalyanpur, A. (2005). Debugging OWL ontologies. In *Proceedings of the World Wide Web Conference (WWW 2005)*. May 10-14, 2005, Chiba, Japan.

[Peñaloza and Turhan, 2011] Peñaloza, R. and Turhan, A.-Y. (2011). A practical approach for computing generalization inferences in EL. In Antoniou, G. et al., editors, *8th Extended Semantic Web Conference (ESWC'11)*, volume 6643 of *LNCS*, pages 410–423. Springer. Heraklion, Crete, Greece, 29 May-2 June, 2011.

[Peroni et al., 2012] Peroni, S., Shotton, D., and Vitali, F. (2012). The Live OWL Documentation Environment: A tool for the automatic generation of ontology documentation. In ten Teije, A. et al., editors, *18th International Conference on Knowledge Engineering and Knowledge Management (EKAW'12)*, volume 7603 of *LNAI*, pages 398–412. Springer. Oct 8-12, Galway, Ireland.

[Pontara, 2011] Pontara, G. (2011). *The nonviolent personality.* page 54. Translated from the Italian original *La personalità nonviolenta.* Online: `http://www.meteck.org/files/NonviolentPersonality.pdf`. Last Accessed: April 15, 2011.

[Portoraro, 2010] Portoraro, F. (2010). Automated reasoning. In Zalta, E., editor, *Stanford Encyclopedia of Philosophy.*

[Poveda-Villalón et al., 2012] Poveda-Villalón, M., Suárez-Figueroa, M. C., and Gómez-Pérez, A. (2012). Validating ontologies with OOPS! In ten Teije, A. et al., editors, *18th International Conference on Knowledge Engineering and Knowledge Management (EKAW'12)*, volume 7603 of *LNAI*, pages 267–281. Springer. Oct 8-12, Galway, Ireland.

[Presutti et al., 2008] Presutti, V., Gangemi, A., David, S., de Cea, G. A., Surez-Figueroa, M. C., Montiel-Ponsoda, E., and Poveda, M. (2008). A library of ontology design patterns: reusable solutions for collaborative design of networked ontologies. NeOn deliverable D2.5.1, NeOn Project, Institute of Cognitive Sciences and Technologies (CNR).

[Queralt and Teniente, 2008] Queralt, A. and Teniente, E. (2008). Decidable reasoning in UML schemas with constraints. In Bellahsene, Z. and Léonard, M., editors, *Proceedings of the 20th International Conference on Advanced Information Systems Engineering (CAiSE'08)*, volume 5074 of *LNCS*, pages 281–295. Springer. Montpellier, France, June 16-20, 2008.

[Randell et al., 1992] Randell, D. A., Cui, Z., and Cohn, A. G. (1992). A spatial logic based on regions and connection. In *Proc. 3rd Int. Conf. on Knowledge Representation and Reasoning*, pages 165–176. Morgan Kaufmann.

[Rector et al., 2004] Rector, A., Drummond, N., Horridge, M., Rogers, L., Knublauch, H., Stevens, R., Wang, H., and Wroe, Csallner, C. (2004). OWL pizzas: Practical experience of teaching OWL-DL: Common errors & common patterns. In *Proceedings of the 14th International Conference Knowledge Acquisition, Modeling and Management (EKAW'04)*, volume 3257 of *LNCS*, pages 63–81. Springer. Whittlebury Hall, UK.

[Rodríguez-Muro and Calvanese, 2012] Rodríguez-Muro, M. and Calvanese, D. (2012). Quest, an OWL 2 QL reasoner for ontology-based data access. In Klinov, P. and Horridge, M., editors, *7th Workshop on OWL: Experiences and Directions (OWLED'12)*, volume 849 of *CEUR-WS*. 27-28 May, Heraklion, Crete, Greece.

[Rodriguez-Muro et al., 2008] Rodriguez-Muro, M., Lubyte, L., and Calvanese, D. (2008). Realizing Ontology Based Data Access: A plug-in for Protégé. In *Proc. of the Workshop on Information Integration Methods, Architectures, and Systems (IIMAS 2008)*. IEEE Computer Society.

[Rosse and Mejino Jr, 2003] Rosse, C. and Mejino Jr, J. L. V. (2003). A reference ontology for biomedical informatics: the foundational model of anatomy. *J. of Biomedical Informatics*, 36(6):478–500.

[Roussey et al., 2009] Roussey, C., Corcho, O., and Vilches-Blázquez, L. (2009). A catalogue of OWL ontology antipatterns. In *Proc. of K-CAP'09*, pages 205–206.

[Rudolph et al., 2008a] Rudolph, S., Krötzsch, M., and Hitzler, P. (2008a). All elephants are bigger than all mice. In Baader, F., Lutz, C., and Motik, B., editors, *Proc. 21st Int. Workshop on Description Logics (DL08)*, volume 353 of *CEUR-WS*. Dresden, Germany, May 1316, 2008.

[Rudolph et al., 2008b] Rudolph, S., Krötzsch, M., and Hitzler, P. (2008b). Cheap boolean role constructors for description logics. In Hölldobler, S., Lutz, C., and Wansing, H., editors, *Proc. 11th European Conf. on Logics in Artificial Intelligence (JELIA08)*, volume 5293 of *LNAI*, page 362374. Springer.

[Sattler, 2007] Sattler, U. (2007). Reasoning in description logics: Basics, extensions, and relatives. In Antoniou, G. et al., editors, *Reasoning Web 2007*, volume 4636 of *LNCS*, page 154182. Springer.

[Schulz et al., 2009] Schulz, S., Stenzhorn, H., Boekers, M., and Smith, B. (2009). Strengths and limitations of formal ontologies in the biomedical domain. *Electronic Journal of Communication, Information and Innovation in Health (Special Issue on Ontologies, Semantic Web and Health)*, 3(1):31–45.

[Schwitter et al., 2008] Schwitter, R., Kaljurand, K., Cregan, A., Dolbear, C., and Hart, G. (2008). A comparison of three controlled natural languages for OWL 1.1. In *Proc. of OWLED 2008 DC*. Washington, DC, USA metropolitan area, on 1-2 April 2008.

[Simperl et al., 2010] Simperl, E., Mochol, M., and Bürger, T. (2010). Achieving maturity: the state of practice in ontology engineering in 2009. *International Journal of Computer Science and Applications*, 7(1):45–65.

[Smith, 2004] Smith, B. (2004). Beyond concepts, or: Ontology as reality representation. In Varzi, A. and Vieu, L., editors, *Formal Ontology and Information Systems. Proceedings of the Third International Conference (FOIS'04)*, pages 73–84. Amsterdam: IOS Press.

[Smith et al., 2007] Smith, B., Ashburner, M., Rosse, C., Bard, J., Bug, W., Ceusters, W., Goldberg, L., Eilbeck, K., Ireland, A., Mungall, C., OBI Consortium, T., Leontis, N., Rocca-Serra, A., Ruttenberg, A., Sansone, S.-A., Shah, M., Whetzel, P., and Lewis, S. (2007). The OBO Foundry: Coordinated evolution of ontologies to support biomedical data integration. *Nature Biotechnology*, 25(11):1251–1255.

[Smith and Ceusters, 2010] Smith, B. and Ceusters, W. (2010). Ontological realism: A methodology for coordinated evolution of scientific ontologies. *Applied Ontology*, 5(3):139–188.

[Smith et al., 2005] Smith, B., Ceusters, W., Klagges, B., Köhler, J., Kumar, A., Lomax, J., Mungall, C., Neuhaus, F., Rector, A. L., and Rosse, C. (2005). Relations in biomedical ontologies. *Genome Biology*, 6:R46.

[Soergel et al., 2004] Soergel, D., Lauser, B., Liang, A., Fisseha, F., Keizer, J., and Katz, S. (2004). Reengineering thesauri for new applications: the AGROVOC example. *Journal of Digital Information*, 4(4).

[Solow, 2005] Solow, D. (2005). *How to read and do proofs: An introduction to mathematical thought processes.* John Wiley & Sons, Hoboken NJ, USA., 4th edition.

[Staab et al., 2001] Staab, S., Schnurr, H., Studer, R., and Sure, Y. (2001). Knowledge processes and ontologies. *IEEE Intelligent Systems*, 16(1):26–34.

[Stocker and Sirin, 2009] Stocker, M. and Sirin, E. (2009). Pelletspatial: A hybrid RCC-8 and RDF/OWL reasoning and query engine. In Hoekstra, R. and Patel-Schneider, P., editors, *Proceedings of the 6th International Workshop OWL: Experiences and Directions (OWLED'09)*, volume 529 of *CEUR-WS*. Chantilly, Virginia, USA, 23-24 October 2009.

[Straccia, 2008] Straccia, U. (2008). Managing uncertainty and vagueness in description logics, logic programs and description logic programs. In *Reasoning Web, 4th International Summer School.*

[Stuckenschmidt et al., 2009] Stuckenschmidt, H., Parent, C., and Spaccapietra, S., editors (2009). *Modular Ontologies—Concepts, Theories and Techniques for Knowledge Modularization.* Springer.

[Studer et al., 1998] Studer, R., Benjamins, R., and Fensel, D. (1998). Knowledge engineering: Principles and methods. *Data & Knowledge Engineering*, 25(1-2):161198.

[Suarez-Figueroa et al., 2008] Suarez-Figueroa, M. C., de Cea, G. A., Buil, C., Dellschaft, K., Fernandez-Lopez, M., Garcia, A., Gómez-Pérez, A., Herrero, G., Montiel-Ponsoda, E., Sabou, M., Villazon-Terrazas, B., and Yufei, Z. (2008). NeOn methodology for building contextualized ontology networks. NeOn Deliverable D5.4.1, NeOn Project.

[Sugumaran and Storey, 2006] Sugumaran, V. and Storey, V. C. (2006). The role of domain ontologies in database design: An ontology management and conceptual modeling environment. *ACM Transactions on Database Systems*, 31(3):1064–1094.

[Tobies, 2001] Tobies, S. (2001). *Complexity Results and Practical Algorithms for Logics in Knowledge Representation.* PhD thesis, RWTH Aachen.

[Turhan, 2008] Turhan, A.-Y. (2008). *On the Computation of Common Subsumers in Description Logics.* PhD thesis, TU Dresden, Institute for Theoretical Computer Science, 2008.

[Turhan, 2010] Turhan, A.-Y. (2010). Reasoning and explanation in EL and in expressive Description Logics. In Assmann, U., Bartho, A., and Wende, C., editors, *Reasoning Web 2010*, volume 6325 of *LNCS*, pages 1–27. Springer.

[van Harmelen et al., 2008] van Harmelen, F., Lifschitz, V., and Porter, B., editors (2008). *Handbook of Knowledge Representation.* Elsevier.

[Varzi, 2007] Varzi, A. (2007). *Handbook of Spatial Logics*, chapter Spatial reasoning and ontology: parts, wholes, and locations, pages 945–1038. Berlin Heidelberg: Springer Verlag.

[Varzi, 2004] Varzi, A. C. (2004). Mereology. In Zalta, E. N., editor, *Stanford Encyclopedia of Philosophy*. Stanford, fall 2004 edition. http://plato.stanford.edu/archives/fall2004/entries/mereology/.

[Varzi, 2012] Varzi, A. C. (2012). On doing ontology without metaphysics. *Philosophical Perspectives*, to appear.

[Vila and Ferrández, 2009] Vila, K. and Ferrández, A. (2009). Developing an ontology for improving question answering in the agricultural domain. In Sartori, F., Sicilia, M., and Manouselis, N., editors, *3rd International Conference on Metadata and Semantics (MTSR'09)*, volume 46 of *CCIS*, pages 245–256. Springer. Oct 1-2 2009 Milan, Italy.

[Vrandečić, 2009] Vrandečić, D. (2009). Ontology evaluation. In Staab, S. and Studer, R., editors, *Handbook on Ontologies*, pages 293–313. Springer, 2nd edition.

[Welty, 2006] Welty, C. (2006). Ontowlclean: cleaning OWL ontologies with OWL. In Bennet, B. and Fellbaum, C., editors, *Proceedings of Formal Ontologies in Information Systems (FOIS'06)*, pages 347–359. IOS Press.

[Witte et al., 2007] Witte, R., Kappler, T., and Baker, C. (2007). Ontology design for biomedical text mining. In Baker, C. and Cheung, H., editors, *Semantic Web: revolutionizing knowledge discovery in the life sciences*, pages 281–313. Springer.

[Wolstencroft et al., 2007] Wolstencroft, K., Stevens, R., and Haarslev, V. (2007). Applying OWL reasoning to genomic data. In Baker, C. and Cheung, H., editors, *Semantic Web: revolutionizing knowledge discovery in the life sciences*, pages 225–248. Springer: New York.

[Zhang et al., 2006] Zhang, S., Bodenreider, O., and Golbreich, C. (2006). Experience in reasoning with the Foundational Model of Anatomy in OWL DL. In Altman, R. B., Dunker, A. K., Hunter, L., Murray, T. A., and Klein, T. E., editors, *Pacific Symposium on Biocomputing (PSB'06)*, pages 200–211. World Scientific.

[Zhou et al., 2005] Zhou, Y., Young, J. A., Santrosyan, A., Chen, K., Yan, S. F., and Winzeler, E. A. (2005). In silico gene function prediction using ontology-based pattern identification. *Bioinformatics*, 21(7):1237–1245.

Books about ontologies

This is a selection of books on ontologies that focus on different aspects of the endeavour; more books are on sale that specialize in a specific subtopic. However, they are generally specialized books presenting relatively new research, or handbooks at best, but none of them is a real textbook (be they suitable for self-study or not). Most of the books are in the UKZN library or online accessible, and more will be added in due time.

Ontology and Ontologies

Steffen Staab and Rudi Studer (Eds.). *Handbook on ontologies.* Springer. 2009.

Zalta (Ed.). *Stanford Encyclopedia of Philosophy.* 2010. `http://plato.stanford.edu/`.

Ontology Engineering

Gomez-Perez, A., Fernandez-Lopez, M., Corcho, O. *Ontological Engineering.* Springer Verlag London Ltd. 2004.

Semantic Web Technologies

Pascal Hitzler, Markus Kroetzsch, Sebastian Rudolph. *Foundations of Semantic Web Technologies*. Chapman & Hall/CRC, 2009, 455p.

John Domingue, Dieter Fensel and James A. Hendler (Eds.). *Handbook of Semantic Web Technologies*. 1st Ed., Springer, 2011, 1056 p.

Various subtopics in ontologies

Jerome Euzenat and Pavel Shvaiko. *Ontology Matching*. Springer. 2007.

Heiner Stuckenschmidt, Christine Parent, Stefano Spaccapietra (Eds.). *Modular Ontologies—Concepts, Theories and Techniques for Knowledge Modularization*. Springer. 2009.

Chu-ren Huang, Nicoletta Calzolari, Aldo Gangemi, Alessandro Lenci, Alessandro Oltramari, Laurent Prevot (Eds.). *Ontology and the lexicon*. Cambridge University Press. 2010.

Ontologies in specific subject domains

Baker, C.J.O., Cheung, H. (Eds). *Semantic Web: revolutionizing knowledge discovery in the life sciences*. Springer: New York, 2007.

Falquet, G., Métral, C., Teller, J., Tweed, C. (Eds.). *Ontologies in Urban Development Projects*. 1st Ed., Springer, 2011, 241 p.

Sartor, G., Casanovas, P., Biasiotti, M., Fernández-Barrera, M. (Eds.). *Approaches to Legal Ontologies—Theories, Domains, Methodologies*. 1st Ed., Springer, 2011, 279 p.

more TBA

General background material

F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider (Eds). *The Description Logics Handbook – Theory and Applications*. Cambridge University Press, 2003/2008.

Halpin, T., Morgan, T. *Information modeling and relational databases*. 2nd Ed. Morgan Kaufmann. 2008.

Frank van Harmelen, Vladimir Lifschitz and Bruce Porter (Eds.). *Handbook of Knowledge Representation*. Elsevier, 2008, 1034p.

Hedman, S. (2004). *A first course in logic—an introduction to model theory, proof theory, computability, and complexity*. Oxford: Oxford University Press.

J. E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Pearson Education, 2nd ed., 2001.

Selection of journals that publish papers about ontologies

Research in ontologies

- Applied Ontology
- International Journal of Metadata, Semantics and Ontologies
- Journal of Web Semantics
- Semantic Web Journal
- Journal on Data Semantics
- Artificial Intelligence Journal
- Journal of Automated Reasoning
- more TBA

Ontologies and applications

- Journal of Biomedical Semantics
- Journal of Biomedical Informatics
- BMC Bioinformatics
- Data & Knowledge Engineering
- more TBA

Selection of conferences that publish papers about ontologies

Research in ontologies

- Formal Ontology in Information Systems (FOIS)
- International Semantic Web Conference (ISWC)
- Extended Semantic Web Conference (ESWC)
- International Workshop on Description Logics (DL)
- International Conference on Knowledge Representation and Reasoning (KR)
- International Conference on Knowledge Engineering and Knowledge Management (EKAW)
- More TBA

Ontologies and applications

- International Conference on Biomedical Ontology (ICBO)
- Semantic Web Applications and Tools for the Life Sciences (SWAT4LS)
- more TBA

APPENDIX A

Practical Assignment: Develop a Domain Ontology

The aim of this practical assignment is for you to demonstrate what you have learned about the ontology languages, top-down and bottom-up ontology ontology development, and methods and methodologies, and experiment with how these pieces fit together.

You can do this assignment in groups of two or three students. It should be mentioned in the material you will hand in who did what.

Tasks

1. Choose a subject domain of interest for which you will develop a domain ontology. For instance, about computers, or for accommodation or tourism websites, furniture, university structures, some hobby (e.g., diving, dancing), or some other subject domain you happen to be knowledgable about (or know someone who is).

2. Develop the domain ontology in the best possible way. You are allowed to use any resource you think is useful, be it other ontologies, non-ontological resources, tools, domain experts, etc.. If you do so (and you are encouraged to do so), then make sure to reference them in the write-up.

3. Write about 2-3 pages (excluding figures or screenshots) summarising your work. This can include—but are not limited to—topics such as an outline of the ontology, why (or why not) you have used a foundational ontology (if so, which, why), if you could reuse a top-domain or other subject domain ontology, which non-ontological resources you have used (if any, and if so, how), if you encountered subject domain knowledge that should have been in the ontology but could not be represented due to the limitations of OWL, or perhaps a (real or imagined) purpose of the ontology and therefore a motivation for some OWL fragment, any particular reasoning services that was useful (and how and why, which deductions did you have or experimented with), any additional tools used.

4. After the deadline, you will review one or two ontologies and write-ups of other groups/students, which we will discuss during the lab after the deadline.

Material to hand in

You have to upload the following items on the course's Vula:

1. The OWL file of your ontology
2. Imported OWL ontologies, if any
3. The write up

Assessment

1. Concerning the ontology: quality is more important that quantity. An ontology with more advanced constraints and appropriate reuse of foundational or general ontologies is a better illustration what you have learned than a large bare taxonomy.
2. Concerning the ontology: it will be checked on modelling errors in the general sense (errors such as is-a vs. part-of, class vs. instance, unsatisfiable classes). Regarding the subject domain itself, it will be checked only insofar as it indicates (mis)understanding of the ontology language or reasoning services.
3. Concerning the write up: a *synthesis* is expected, not a diary. For instance, "We explored a, b, and c, and b was deemed to be most effective because blabla" would be fine, but not "We tried a, but that didn't work out, then we had a go at b, which went well, then we came across c, tried it out of curiosity, but that was a dead end, so we went back to b.". In short: try to go beyond the 'knowledge telling' and work towards the so-called *knowledge transformation*.
4. Concerning the write up: while a brief description of the contents is useful, it is more important to include something about the *process* and *motivations* how you got there, covering topics such as, but not limited to, those mentioned under Tasks, item 3 (and recollect the aim of the assignment—the more you demonstrate it, the better).

Notes

In random order:

1. The assignment looks easy. It isn't. If you start with the development of the ontology only the day or so before the deadline, there is an extremely high probability that you will fail this assignment. Your assignment will be of a higher quality if you start thinking about it some 2 before the deadline, and the actual development at most one week before the deadline, and spread out the time you are working on it.
2. If you use non-English terms for the classes and properties, you should either add the English in the annotations (preferred), else lend me a dictionary.
3. Use proper referencing when you use something from someone else, be it an ontology, other reused online resources (including uncommon software), textbooks, articles etc. Not doing so amounts to plagiarism; see Appendix C for guidelines.
4. Spell checkers tend to be rather useful tools.
5. Some of the mini-project topics can benefit from an experimental ontology that you know in detail, which you may want to take into consideration when choosing a subject domain or purpose so that your ontology might be reused later on.

APPENDIX B

Assignment: Mini-project

The aim of this assignment is to work in a small group (204 studnets) and to investigate a specific theme of ontology engineering. The topics are such that either you can demonstrate the *integration of various aspects* of ontologies and knowledge bases or *going into quite some detail on a single topic*, and it can be either theory-based, implementation-focussed, or a bit of both.

Possible topics to choose from will be communicated in week 2, and has to be chosen and communicated to me (topic + group members) no later than in week 3, else you will be assigned a topic and a group.

Tasks

1. Form a group of 2-4 people and choose a topic, or vv.: choose a topic and find other people to work with. It should be mentioned in the material you will hand in who did what.
2. Carry out the project.
3. Write about 4-6 pages (excluding figures or screenshots) summarising your work. The page limit is flexible, but it surely has to be $<$ 15 pages in total.
4. Give a presentation of your work during the last lecture (10 minutes presentation, $\pm$5 minutes discussion). *Everyone must attend this lecture.*

Material to hand in

You have to upload the following items on the course's Vula site:

1. The write up.
2. Additional material: this depends on the chosen topic. If it is not purely paper-based, then the additional files have to be uploaded on the system (e.g., software, test data).
3. Slides of the presentation, if any.

Note that the deadline is after the last lecture so that you have the option to update your material for the mini-project with any feedback received during the presentation and discussion in class.

Assessment

1. Concerning the write up: a *synthesis* is expected, not a diary. For instance, "We explored a, b, and c, and b was deemed to be most effective because blabla" would be fine, but not "We tried a, but that didn't work out, then we had a go at b, which went well, then we came across c, tried it out of curiosity, but that was a dead end, so we went back to b.". In short: try to go beyond the 'knowledge telling' and work towards the so-called *knowledge transformation*.
2. Concerning the write up: use proper referencing when you use something from someone else, be it an ontology, other reused online resources (including uncommon software), textbooks, articles etc. Not doing so amounts to plagiarism, which has a minimum penalty of obtaining a grade of 0 (zero) for the assignment (for all group members, or, if thanks to the declaration of contribution the individual can be identified, then only that individual), and you will be recorded on the departmental plagiarism list, if you are not already on it, and further steps may be taken. See also Appendix C for guidelines on referencing material.
3. The presentation: respect the time limit, coherence of the presentation, capability to answer questions.
4. Concerning any additional material (if applicable): if the software works as intended with the given input, presentability of the code.
5. Marks will be deducted if the presentation or the write-up is too long.

Notes

Things you may want to take into consideration (listed in random order):

1. LaTeX is a useful typesetting system (including beamer for presentation slides), has a range of standard layouts as well as bibliography style files to save you the trouble of wasting time on making the write up presentable, and generates a pdf file that is portable[1]. This is much less so with MS Word; if you use MS Word nevertheless, please also include a pdf version of the document.
2. Regarding the bibliography: have complete entries. Examples of referencing material (for conference proceedings, books and book chapters, and journal articles) can be found in the scientific papers included in the lecture notes' bibliography, scientific literature you consult, and Appendix C.
3. Spell checkers tend to be rather useful tools.
4. One or more of the domain ontologies developed in the previous assignment may be suitable for reuse in your chosen topic.
5. Some of the mini-projects lend themselves well for extension into an Honours project, hence, could give you a head-start.

[1]in case you are not convinced: check `http://openwetware.org/wiki/Word_vs._LaTeX` or `http://ricardo.ecn.wfu.edu/~cottrell/wp.html`

APPENDIX C

Dealing with References

C.1 Introduction

When working on a project or a thesis, one may, at times, feel as if one is working alone in isolation. This may be true for various reasons, but ought not to be the case, unless you are working on a PhD thesis when you have to demonstrate you have what it takes to go at least one whole turn through the knowledge creation spiral. What certainly holds, however, is that

1. the *topic* of your project or thesis does not exist in isolation, and
2. you are supposed to reinvent the wheel.

Put differently: *you are building upon other people's contributions.* That is how science and engineering move forward.

Building upon other people's results entails that you are aware what they did and understand the pros and cons of those earlier proposals. This *must* be reflected in the article or thesis you are or will be writing. In this document, we focus on how to go about giving credit where credit is due; not doing so amounts to **plagiarism**, which is the sin of sins in academia, on a par with inventing data and modifying them (that also constitutes scientific misconduct). If you plagiarise during your thesis writing, you will be reprimanded officially, if you have done so in your thesis and it is found out after graduation, your degree will be revoked, if you do it as a scientist, you will be fired. If you do so in a write up for an assignment of this module and I find out, you will receive a 0, a warning, and you are required to give a statement you wont do it again; if I catch you a second time, you will fail the module and the case will be recorded in the university system (note: given the weighting of the components that make up the grade, with one 0 you are very likely to fail the module already anyway).

C.1.1 Plagiarism and borderlines

Plagiarism is to copy whole sentence(s), figures, tables—any material—from another document without giving a reference to the source you took it from, be this someone else's or your own. 'Copy' here is taken liberally. So, if you were to take the original sentence from [4]

On average, those who commenced with a foundational ontology added more

> new classes and class axioms, and significantly less object properties than those who started from scratch.

and copy it without putting it in indentation as quote or between quotation marks (“”), it is plagiarism. Or when you make only very minor changes, e.g.,

> On average, those who started with a foundational ontology added more new classes and class axioms, and significantly less object properties than those who started with a *tabula rasa*.

without adding a reference to [4], then it is still plagiarism, even though the two strings are not exactly the same. Note that when you find useful information online that you use in your writings, you have to reference that. It may not always be clear who provided that information and what the exact publication date was, or it may take some effort to find out, but it wasn't yours, so you are not allowed to claim as if it were.

This is difficult for mathematically-oriented papers when one has to introduce the syntax and semantics of a language that was already presented in another paper, so there are exceptions for those paragraphs (but try to fiddle a bit, where possible), provided the reference to the paper that introduced the language is given (e.g., [2] reuses the $\mathcal{DLR}_{\mathcal{US}}$ language introduced in [1], duly referenced).

This raises a new question: how to cite your source? This appendix is not a long list about the different referencing styles for the different type of documents—there is software who can handle that for you—but instead it contains a few guidelines on how you can refer to the references, the references section itself, managing references, and what is (not) referencable. (To reiterate: a section with references is *essential* to the report/article/thesis.)

C.2 Referencing in text

I will introduce first a difference between quoting and citing material, which also gives a first example of a reference in the text. There are two principle *ways of referencing* in the text: footnote/endnote or names/numbers, and, within each option, there are minor style variations; either way you choose: **be consistent** throughout the text.

C.2.1 Quoting versus citing

Longer quotes, like the one from [4] in Section C.1, should be *indented*, whereas shorter pieces can be put in-line in quotation marks, e.g.:

> We are interested only in the “significantly less object properties” of [4] instead of the whole gamut.

You also can emphasise something in a quotation, but this has to be indicated as such. Here is an example of a longer, indented quotation and emphasis:

> Against expectations, Keet [4] concluded the following from her experiments:
>
> > On average, those who commenced with a foundational ontology added more new classes and class axioms, and *significantly less object properties* than those who started from scratch. (emphasis added)
>
> Now, let us take a look at that peculiar aspect of it.

Where possible, add also the page number to the reference.

Quotations have to be truthful to the original. For instance, if the original sentence were:

> Ontologists tend to be outspoken about the usefulness of foundational (top-level) ontologies, such as BFO, DOLCE [1], GFO [2], and SUMO: either they are perceived to be essential or an impractical burden.

and you quote it as:

> Despite our own experiences in developing domain ontologies, Keet [4] dares to claim that "... foundational (top-level) ontologies ... are ... essential."

then this is clearly *not* truthful to the original sentence. It is permitted to skip *irrelevant* parts of a larger piece of text as long as they do not modify the overall message of the original.

Fiddling a bit with the layout to correct style (captialization) or substituting a relative pronoun with the original noun is permitted, which is always put between square brackets; for instance:

> The dispute was succinctly summarised as "either [foundational ontologies] are perceived to be essential or an impractical burden" [4].

Instead of a quotation, which is fairly common in the humanities but not in computer science (because it takes up a lot of space), once can paraphrase it. For instance, you can put in the text

> Keet [4] demonstrated that the 52 test subjects added, on average, more new OWL classes and class axioms, and significantly less object properties when they started with the OWLized DOLCE.

or

> It has been shown that more new classes and class axioms were added when ontology developers started with a foundational ontology compared to those who did not [4].

In these two cases you *cite* material, but do not *quote* it. You put the same message in your own words with such citations: the fact or idea is not new, and hence the reference where it originates, but it is not taken verbatim from that source.

C.2.2 Referencing modes

The first way of referencing, footnote or endnote, is very common in the humanities when it comes to 'real' references. For computer science, the may be common for Web references *only*, and even then it depends on the outlet which one is preferred (and due to page limitations, one may take up less space than the other, hence make the difference). An example of footnote referencing is demonstrated the following text passage:

> What constitutes a nonviolent personality? Pontara describes 10 characteristics that a person with a nonviolent personality should posess to a great extent[1]. One

[1]Pontara, G. (2011). The nonviolent personality. In: Keet, C.M. (Ed.), translated from Italian *La personalità nonviolenta*. 54p. Online: `http://www.meteck.org/files/NonviolentPersonality.pdf`. Last accessed: April 15, 2011.

> of these is mildness, where he asserts in §3.2.7 that "[a] person equipped with a nonviolent personality is not outside the political struggle"[2], which I will discuss in detail in the remainder of this essay.

That is, the first reference to the document written by Pontara [6] is referenced in full, the subsequent one refers to the previous note and contains only the page number. The full citation is normally also included in a separate "References" section at the end of the text.

Referencing with footnotes occurs also in computer science literature, but this is limited to online sources, and is not nice behaviour toward the originators when a proper article of the online source exist (references count in citation indices—the more your paper is cited, the higher the impact of your contribution to the field—but footnotes/endnotes do not count). For instance, it is possible to do the following

> We developed our ontology in Protégé 4.1beta[3] by first importing the taxonomy of part-whole relations[4].

Very nice would be, instead:

> We developed our ontology in Protégé 4.1beta [3] by first importing the taxonomy of part-whole relations [5].

However, citing software is, to a large extent, a judgement call and depends on how widely known the software is. Protégé is so widely known within the ontology development community so that, if you write for *that* community, one would not cite it anymore unless tehre is something special about it. Surely one would not cite—article or URL—say, Linux OS in your materials & methods section (unless there's a very recent flashing new earth-shattering feature to be highlighted). New, or for the target audience unfamiliar, software applications should be cited so that a reader can follow up on what you have done.

You have already come across the other referencing mode, because it has been used throughout this appendix. More specifically, I used the numbering scheme where a number (that referred to a reference) was inserted in the main text, like the "... Protégé 4.1beta [3] by..." in the previous example. There is just one section of references, which are numbered alphabetically in this case, and that's it. The previous example with a naming scheme instead of numbers could look like this:

> We developed our ontology in Protégé 4.1beta [GMF$^+$03] by first importing the taxonomy of part-whole relations [KA08].

or one can use the full names instead of an abbreviation:

> We developed our ontology in Protégé 4.1beta (Gennari *et. al.*, 2003) by first importing the taxonomy of part-whole relations (Keet & Artale, 2008).

The final display of the in-text references depends on which *referencing style* you use, which will receive attention in Section C.4.

[2] *op. cit.* p27.

[3] `http://stanford.protege.edu`

[4] `http://www.meteck.org/files/ontologies/pwrelations.owl`

C.2.3 What to put where

The examples in the previous section already introduced several options for how to include a reference in the text. Here I shall list that more systematically.

What has most effect on the formatting is if you choose a numbering or a naming scheme. For instance, this is good:

> Keet and Artale (2008) developed a taxonomy of part-whole relations.

but if the reference were a number, rendering the sentence like this:

> [5] developed a taxonomy of part-whole relations.

then that is not acceptable. In the second case, you would have to write

> Keet and Artale [5] developed a taxonomy of part-whole relations.

or, if you refer to more details, then the reference can be put at the end of the sentence:

> Keet and Artale developed a taxonomy of part-whole relations, which has as major division between 'real' parthood relations versus those relations that are motivated by linguistics and might seem a parthood relation, but are not [5].

or you can skip the author(s)' names, and write it in a more detached way:

> The most recently proposed taxonomy of part-whole relations has as major division between 'real' parthood relations versus those relations that are motivated by linguistics and might seem a parthood relation, but are not [5].

When using the naming scheme and there are more than three authors for a single reference, you do not include the whole list of authors in the text, but abbreviate it with *et. al.*—in italics, and with the dots. We have seen this in an earlier example already, e.g., with the reference to Protégé:

> ... ontology in Protégé 4.1beta (Gennari *et. al.*, 2003) by first...

Some software can do this automatically for you.

You also can combine references into one list, where appropriate. For instance,

> ... with some recent results on part-whole relations that also build upon temporal logics in general [1, 2, 5].

instead of the not incorrect but unpleasantly looking

> ... with some recent results on part-whole relations that also build upon temporal logics in general [1], [5], [2].

Observe two things in the example: the neater version has all the references within the same square brackets, and in numerical order. The latter is not required, but neater.

C.3 The "References" section

The list of references goes at the end of the text. There is no consistency to do that always after the main text but before the appendix, or also after the appendix. Check other theses/papers for that, or, if you have an official template, adhere to the template. Regardless where it ends up after the text, the references in the "References" section have to be **consistent** and **complete**.

Whichever referencing style you use, *do the layout the same for the same type of entries*: if you do book titles in italics, then do so for all of them, if you include the total amount of pages of a book, do so for all of them, and so forth.

They also have to be complete to the extent that a reader should be able to find that paper. For instance, [3, 5] are complete entries. Conference and journal abbreviations and cutting down a long list of authors to one or three authors with an "*et. al.*" may be acceptable, too (and in some outlets even required), e.g.:

> Gennari, J. H. *et. al.*. The evolution of Protégé: an environment for knowledge-based systems development. *Int. J. of Hum.-Comp. Studies 58*, 1 (2003), 89-123.

may be fine if you run out of space, but *not* something like

> Gennari, J. H. *et. al.*. The evolution of Protégé: an environment for knowledge-based systems development. 2003.

because it misses the publication venue details, nor

> Gennari, J. H., Musen, M. A., Fergerson, R. W., Grosso, W. E., Crubézy, M., Eriksson, H.. *Int. J. of Hum.-Comp. Studies*, 1 (2003).

because it misses the title of the paper, volume of the journal and page numbers, and has only several authors instead of all (or: misses the "*et. al.*" after the last-mentioned author). A comprehensive version of this and other references used in this appendix can be found at the end of this appendix in the references section.

References to conference papers have their own acceptable or tolerable 'shorthand notations'. For instance, if the authors were really short on space, then a complete reference like [1] can be seen to be abbreviated as:

> Artale, A., Franconi, E., Wolter, F., and Zakharyaschev, M. A temporal description logic for reasoning about conceptual schemas and queries. *Proc. of JELIA'02*, Springer LNAI 2424, 98-110.

But again, this is only by exception and is done only when the readership is familiar with those venues. You, as a budding scientist reading those papers, may not be—and that is the only reason why the two examples of 'condensed' references are described here. You are not short on space, and have to give full details of the reference.

C.3.1 Finding the details of those 'random' online documents

There are many documents on the web and when you have search for information on, e.g. Google Scholar, clicked a pdf, then it might seem that is all there is. This is not the case, especially since computer scientists generally archive their papers on their home page—just that that document does not have all the citation data does not mean it has not been published. So for instance, the following practice is wrong:

... One of the semantic wikis is SweetWiki [1] that uses RDF [2]. ...

References
1. http://smtp.websemanticsjournal.org/index.php/ps/article/viewFile/138/136
2. http://www.semantic-web-book.org/w/images/7/73/Informatik09-Semantic-Web-1-RDF.pdf

The *real* references for [1] and [2] that should appear in your references section are as follows:

1. Buffa, M., Gandonb, F., Ereteo, G. Sander, P., Faron. C. SweetWiki: A semantic wiki. *Journal of Web Semantics*, 2008, **6**(1): 84-97.
2. Hitzler, P., Kroetzsch, M., Rudolph, S. *Foundations of Semantic Web Technologies*. Chapman & Hall/CRC, 2009, 455p.

The trick is to find out that the first URL is actually pointing to a a preprint version of a published scientific paper and the second URL to tutorial slides of a textbook book (which can be referenced provided you did read the book).

The quick and dirty way to find out is to put the title of the document in Google Scholar. If it finds it, then look at the bottom row of the hit, which has hyperlinks like 'cited by', and also a 'cite'. Click on 'cite' and it gives you the reference in three different styles. Pick one and put it in the references; considering the gradations of (in)completeness, that is definitely better than just the URL. However, those automatically generated references are somewhat incomplete, especially when it comes to published conference papers. Another option is to look up the publications page of one of the paper's authors, and take the data from there. If you have figured out the publisher, then you also can go to the publisher's site to obtain the details. Finally, a lot of computer science papers have been indexed in the computer science bibliography server at `http://www.informatik.uni-trier.de/~ley/db/index.html`, which can generate the reference for you. If that does not work out well, like with the second URL: try to understand the URL. A "semantic-web-book" in it is a pretty obvious clue.

C.4 Managing your references

Nevertheless, this raises the question: What are the 'minimal' components of a reference? This depends on the reference *style* you use. For instance, there is a so-called "Harvard referencing" and "Chicago style" that have elaborate guides with the dos and don'ts, there are common styles in computer science, such as of the IEEE, Springer LNCS, and ACM, and journals tend to have their own requirements how to reference each type of resource. You'd be crazy to read through all those guidelines and adapt your references each time you have to follow another style. Scientists and software developers got together, and developed reference management software to do this for you, so, unlike other resources on referencing (and there are a lot of them), I will not waste time re-writing a guide on that, because we can get that sorted out automatically.

When you conduct your literature research to read up on the topic you have chosen for your thesis, article, or technical report, you will come across many resources, some more useful than others. Of those deemed relevant, you should note the publication details immediately to save yourself from wasting time searching for that resource again in a few months' time. What information about the publication venue of that resource should you record? We have seen in the previous section that the notion of 'complete' information might, in fact, vary slightly. In addition, you may have noticed that the references you have come across have different layouts. Although you are going to focus on writing your honours project now, you may wish to continue with a masters and/or, if the results obtained are really good and novel, you may be involved in writing a scientific paper afterwards, which may have its own requirements for formatting references. Manually reformatting the references is a painstaking task and prone to introduce inconsistencies in the layouts. The latter is considered *sloppyness* and not appreciated: if you obviously do not care about your text, why should the reader spend precious time to read it?

Fortunately, many have gone before you who wanted to have a nice software-based reference management system that also does automatically the reformatting of the references adjusted to the venue and in the right order. There are now several such tools around that come in three different flavours. First, there are 'general science' reference management tools for people who write texts in OpenOffice or MS Word, such as Reference Manager[5] and Endnote[6], which, however, are stand-alone applications that are not for free. Second, there are free tools that take a social networking approach to reference management, such as Mendeley[7]. The third one is for LaTeX aficionados, such as Jabref[8], BibDesk[9], and BibTool[10] (more general information at: `http://www.bibtex.org/`).

Basically, you store your references in a fancy database and each time you use a reference, you insert its key in the text. Once ready, the selected keys, your text editor, and your 'selected export format' get together and produce the right amount of references in the right format in the right order—automatically.

Let's have a quick look at an example for LaTeX, which is the preferred editing and typesetting program of computer scientists. Your text goes in a `.tex` file, your references go in a `.bib` file, and your choice for reference format is a selected `.bst` file. At the end of your main file, say, `myproject.tex` you add two lines, one to specify the reference *style*, say, `model3-num-names.bst`, and one to refer to your bibliography, say `mybib.bib`, which then looks like this:

```
\bibliographystyle{model3-num-names}
\bibliography{mybib}
```

You then have to run latex-bibtex-latex-latex to get all references fully resolved. To produce the references for this document, I used the `acm.bst` format, and there are very many more options available that come both with a LaTeX editor distribution and online. Have a look at, e.g., this resource[11]. For instance, when I use the very same bib file but with

```
\bibliographystyle{abbrvnat}
\bibliography{mybib}
```

the references look like depicted in Figure C.1. Not just that, but you would not have seen numbers in square brackets in this document, but the authors' names, like:

> ...importing the taxonomy of part-whole relations [Keet and Artale(2008)].

You may not like the square brackets, or want to have the in-text format alike "(Keet & Artale, 2008)" instead of "[Keet and Artale(2008)]". The `.bst` files are customizable,

[5]`http://www.refman.com/`
[6]`http://www.endnote.com/`
[7]`http://www.mendeley.com/`
[8]`http://jabref.sourceforge.net/`
[9]`http://bibdesk.sourceforge.net/`
[10]`http://www.gerd-neugebauer.de/software/TeX/BibTool/index.en.html`
[11]`http://www.cs.stir.ac.uk/~kjt/software/latex/showbst.html`

[Artale et al.(2002)Artale, Franconi, Wolter, and Zakharyaschev] A. Artale, E. Franconi, F. Wolter, and M. Zakharyaschev. A temporal description logic for reasoning about conceptual schemas and queries. In S. Flesca, S. Greco, N. Leone, and G. Ianni, editors, *Proceedings of the 8th Joint European Conference on Logics in Artificial Intelligence (JELIA-02)*, volume 2424 of *LNAI*, pages 98–110. Springer Verlag, 2002.

[Artale et al.(2008)Artale, Guarino, and Keet] A. Artale, N. Guarino, and C. M. Keet. Formalising temporal constraints on part-whole relations. In G. Brewka and J. Lang, editors, *11th International Conference on Principles of Knowledge Representation and Reasoning (KR'08)*, pages 673–683. AAAI Press, 2008. URL `http://www.meteck.org/files/AGK_KR08.pdf`. Sydney, Australia, September 16-19, 2008.

[Gennari et al.(2003)Gennari, Musen, Fergerson, Grosso, Crubézy, Eriksson, Noy, and Tu] J. H. Gennari, M. A. Musen, R. W. Fergerson, W. E. Grosso, M. Crubézy, H. Eriksson, N. F. Noy, and S. W. Tu. The evolution of Protégé: an environment for knowledge-based systems development. *International Journal of Human-Computer Studies*, 58(1):89–123, 2003. ISSN 1071-5819. doi: http://dx.doi.org/10.1016/S1071-5819(02)00127-1.

[Keet(2011)] C. M. Keet. The use of foundational ontologies in ontology development: an empirical assessment. In G. Antoniou et al., editors, *8th Extended Semantic Web Conference (ESWC'11)*, volume 6643 of *LNCS*, pages 321–335. Springer, 2011. Heraklion, Crete, Greece, 29 May-2 June, 2011.

[Keet and Artale(2008)] C. M. Keet and A. Artale. Representing and reasoning over a taxonomy of part-whole relations. *Applied Ontology – Special issue on Ontological Foundations for Conceptual Modeling*, 3(1-2):91–110, 2008. URL `http://www.meteck.org/files/AO07_pw_AK07.pdf`.

[Pontara(2011)] G. Pontara. *The nonviolent personality.* page 54. March 2011. URL `http://www.meteck.org/files/NonviolentPersonality.pdf`. Translated from the Italian original *La personalità nonviolenta*. Online: http://www.meteck.org/files/NonviolentPersonality.pdf. Last Accessed: April 15, 2011.

Figure C.1: The same references, but formatted with `abbrvnat`.

but before you have a go at that, you may want to check out the very powerful `natbib` package that already does most of this for you.

C.5 What you can (not) cite

Your thesis/article/tech report is a serious academic document. This means that you use, and build upon—hence, also reference—proper academic and reliable, material of good quality. As a rule of thumb:

- What you can reference: journal articles, conference papers, books, chapters in books (including edited conference proceedings), technical reports, other theses, standards. In general: they are *primary sources*.
- What you should *not* reference (under normal circumstances, in CS): blogs and blog posts, posts on forums, Joe Soap's tips 'n tricks. These are, at best, *secondary sources*.

Web pages and company product manuals are trickier, but, in general, they should be avoided whenever possible. For instance, a 'white paper' by Company of Ruby And Perl on the Fabulously Implemented Algorithm to SCam Others may claim that their tool is the coolest around, but any company will say that about their applications. Let "[7]" be the reference to that online white paper, then you might be tempted to write a sentence in your thesis alike

CRAP's FIASCO has the best performance [7].

A press release may say so, but your thesis is not a mouthpiece for the company. If, on the other hand, there is a paper *independently* from the organisation that compared it to similar tools and has shown it experimentally to hold, then that resource should be referenced.

Another tricky resource is Wikipedia, because it provides nice introductory overviews of many topics. However, Wikipedia is a secondary source and you should have a (reliable) textbook or handbook on those topics, in particular when it comes to your thesis. In this case, it is appropriate to reference the textbook.

Regarding blogs, while they should be avoided as reference, if you find, say, a piece of *good* code that someone put on their blog and it cannot be found elsewhere, it is better to reference that than claiming you invented it when you did not.

Chapter References

[1] Artale, A., Franconi, E., Wolter, F., and Zakharyaschev, M. A temporal description logic for reasoning about conceptual schemas and queries. In *Proceedings of the 8th Joint European Conference on Logics in Artificial Intelligence (JELIA-02)* (2002), S. Flesca, S. Greco, N. Leone, and G. Ianni, Eds., vol. 2424 of *LNAI*, Springer Verlag, pp. 98–110.

[2] Artale, A., Guarino, N., and Keet, C. M. Formalising temporal constraints on part-whole relations. In *11th International Conference on Principles of Knowledge Representation and Reasoning (KR'08)* (2008), G. Brewka and J. Lang, Eds., AAAI Press, pp. 673–683. Sydney, Australia, September 16-19, 2008.

[3] Gennari, J. H., Musen, M. A., Fergerson, R. W., Grosso, W. E., Crubézy, M., Eriksson, H., Noy, N. F., and Tu, S. W. The evolution of Protégé: an environment for knowledge-based systems development. *International Journal of Human-Computer Studies 58*, 1 (2003), 89–123.

[4] Keet, C. M. The use of foundational ontologies in ontology development: an empirical assessment. In *8th Extended Semantic Web Conference (ESWC'11)* (2011), G. Antoniou et al., Eds., vol. 6643 of *LNCS*, Springer, pp. 321–335. Heraklion, Crete, Greece, 29 May-2 June, 2011.

[5] Keet, C. M., and Artale, A. Representing and reasoning over a taxonomy of part-whole relations. *Applied Ontology – Special issue on Ontological Foundations for Conceptual Modeling 3*, 1-2 (2008), 91–110.

[6] Pontara, G. *The nonviolent personality.* March 2011, p. 54. Translated from the Italian original *La personalità nonviolenta.* Online: http://www.meteck.org/files/NonviolentPersonality.pdf. Last Accessed: April 15, 2011.

Additional sources

Coxhead, Peter. A Referencing Style Guide. University of Birmingham. Last updated: 16 Nov 2009. `http://www.cs.bham.ac.uk/~pxc/refs/index.html` Last accesses: April 15, 2011.

The Open University. The IT and Computing Project—Resources. Faculty of Technology. version 2. 2003 [CD-ROM CDR0579].

APPENDIX D

Answers of selected exercises

D.1 Answers Chapter 2

Answer to Exercise 1.
A: Dalila travels to Johannesburg
B: Dalila travels to Durban
C: Dalila takes the plane
Then we can formalise the three sentences, above as: $((A \vee B) \wedge (A \rightarrow C)) \rightarrow \neg B$
Truth table: satisfiable.

```
A   B C (((A v B) ^ (A => C)) => ~ B)
T   T T F
T   T F T
T   F T T
T   F F T
F   T T F
F   T F F
F   F T T
F   F F T
```

Answer to Exercise 2.
(b) There exists a node that does not participate in an instance of R, or: it does not relate to anything else: $\exists x \forall y. \neg R(x, y)$.

(c) $\mathcal{L} = \langle R \rangle$ as the binary relation between the vertices. Optionally, on can add the vertices as well. Properties:
R is symmetric: $\forall xy. R(x, y) \rightarrow R(y, x)$.
R is irreflexive: $\forall x. \neg R(x, x)$.
If you take into account the vertices explicitly, one could say that each note participates in at least two instances of R to different nodes.

Answer to Exercise 3.
(a) R is reflexive (a thing relates to itself): $\forall x. R(x, x)$.
R is asymmetric (if a relates to b through relation R, then b does not relate back to a

through R): $\forall xy.R(x,y) \rightarrow \neg R(y,x)$.

(b) See the example on p21 of the lecture notes.

D.2 Answers Chapter 3

Answer to Exercise 6.
(a) Rewrite (Eq. 3.24) into negation normal form:
$Person \sqcap \forall eats.Plant \sqcap (\neg Person \sqcup \neg\forall eats.(Plant \sqcup Dairy))$
$Person \sqcap \forall eats.Plant \sqcap (\neg Person \sqcup \exists\neg eats.(Plant \sqcup Dairy))$
$Person \sqcap \forall eats.Plant \sqcap (\neg Person \sqcup \exists eats.(\neg Plant \sqcap \neg Dairy))$
So our initial ABox is:
$S = \{(Person \sqcap \forall eats.Plant \sqcap (\neg Person \sqcup \exists eats.(\neg Plant \sqcap \neg Dairy)))(a)\}$

(b) Enter the tableau by applying the rules (see lecture slide 31) until either you find a completion or only clashes.
($\sqcap$-rule): $\{Person(a), \forall eats.Plant(a), (\neg Person \sqcup \exists eats.(\neg Plant \sqcap \neg Dairy))(a)\}$
($\sqcup$-rule):
(1) $\{Person(a), \forall eats.Plant(a), (\neg Person \sqcup \exists eats.(\neg Plant \sqcap \neg Dairy))(a), \neg Person(a)\}$ ¡clash!
(2) $\{Person(a), \forall eats.Plant(a), (\neg Person \sqcup \exists eats.(\neg Plant \sqcap \neg Dairy))(a), \exists eats.(\neg Plant \sqcap \neg Dairy)(a)\}$
($\exists$-rule): $\{Person(a), \forall eats.Plant(a), (\neg Person \sqcup \exists eats.(\neg Plant \sqcap \neg Dairy))(a), \exists eats.(\neg Plant \sqcap \neg Dairy)(a), eats(a,b), (\neg Plant \sqcap \neg Dairy)(b)\}$
($\sqcap$-rule): $\{Person(a), \forall eats.Plant(a), (\neg Person \sqcup \exists eats.(\neg Plant \sqcap \neg Dairy))(a), \exists eats.(\neg Plant \sqcap \neg Dairy)(a), eats(a,b), (\neg Plant \sqcap \neg Dairy)(b), \neg Plant(b), \neg Dairy(b)\}$
($\forall$-rule): $\{Person(a), \forall eats.Plant(a), (\neg Person \sqcup \exists eats.(\neg Plant \sqcap \neg Dairy))(a), \exists eats.(\neg Plant \sqcap \neg Dairy)(a), eats(a,b), (\neg Plant \sqcap \neg Dairy)(b), \neg Plant(b), \neg Dairy(b), Plant(b)\}$ ¡clash!

(c) $\mathcal{T} \vdash Vegan \sqsubseteq Vegetarian$? yes

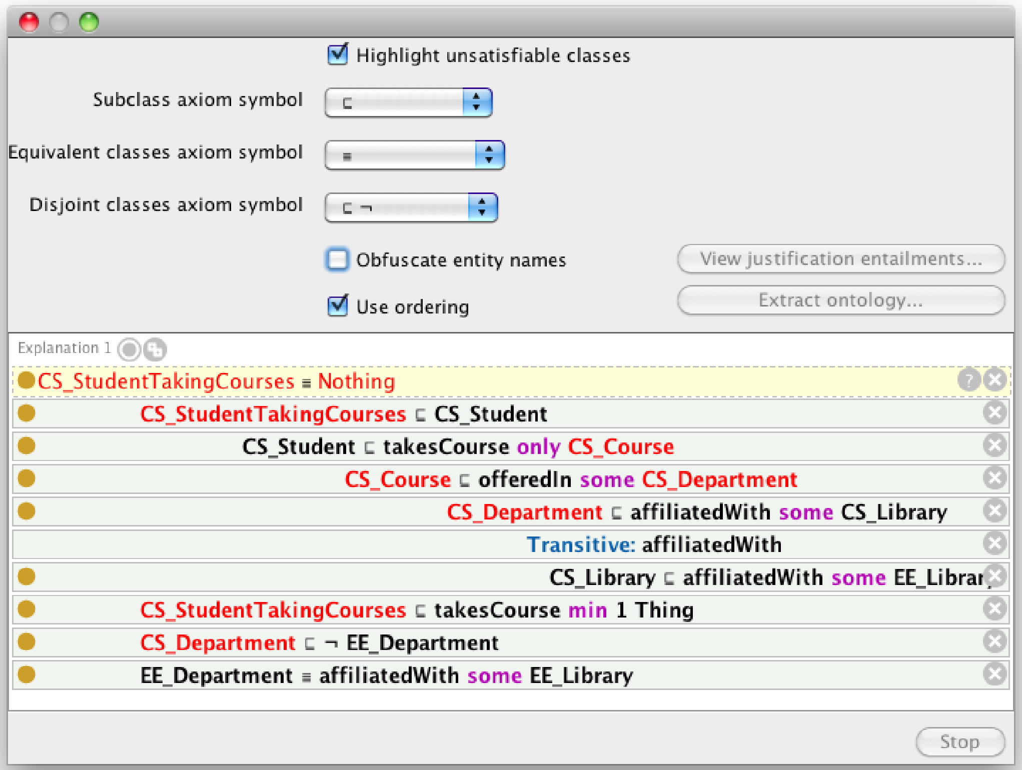
D.3 Answers Chapter 4

Answer to Exercise 14.
Hint: use a property chain.

Answer to Exercise 15.
Modelling: The answers and considerations for the `university1.owl` exercises can be found at `http://owl.man.ac.uk/2005/07/sssw/university.html`, and note that those answers hold for OWL-DL. Does it make a difference with OWL 2 DL? E.g., would the new feature of qualified cardinality constraints be of any use? Not really.

Answer to Exercise 16. to 18
See `http://owl.man.ac.uk/2005/07/sssw/university.html`. We will discuss this in the lab.

Answer to Exercise 20. and 20
Let us first have a look randomly at a deduction and its explanation (click on the "?" right from the deduction in Protégé) as a first step toward figuring out why so many classes are unsatisfiable (i.e., equivalent to `Nothing`, or $\perp$). Take the explanation for `CS_StudentTakingCourses`:



☑ Highlight unsatisfiable classes

Subclass axiom symbol: ⊑

Equivalent classes axiom symbol: ≡

Disjoint classes axiom symbol: ⊑ ¬

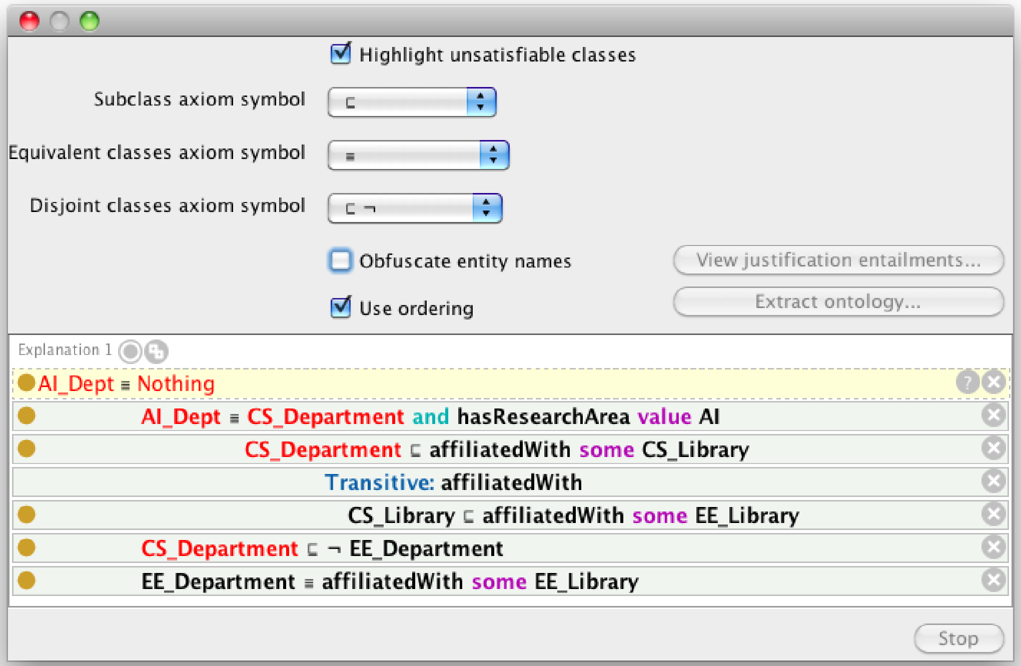
☐ Obfuscate entity names

☑ Use ordering

```
● CS_StudentTakingCourses ≡ Nothing
●     CS_StudentTakingCourses ⊑ CS_Student
●         CS_Student ⊑ takesCourse only CS_Course
●             CS_Course ⊑ offeredIn some CS_Department
●                 CS_Department ⊑ affiliatedWith some CS_Library
                      Transitive: affiliatedWith
●                     CS_Library ⊑ affiliatedWith some EE_Librar
●     CS_StudentTakingCourses ⊑ takesCourse min 1 Thing
●     CS_Department ⊑ ¬ EE_Department
●     EE_Department ≡ affiliatedWith some EE_Library
```

This CS_StudentTakingCourses has a long explanation of why it is unsatisfiable, and we see that some of the axioms that it uses to explain the unsatisfiability also have unsatisfiable classes. Hence, it is a good idea to set this aside for a while, as it is a knock-on effect of the others that are unsatisfiable.

Let us have a look at the unsatisfiability regarding departments.



☑ Highlight unsatisfiable classes

Subclass axiom symbol: ⊑

Equivalent classes axiom symbol: ≡

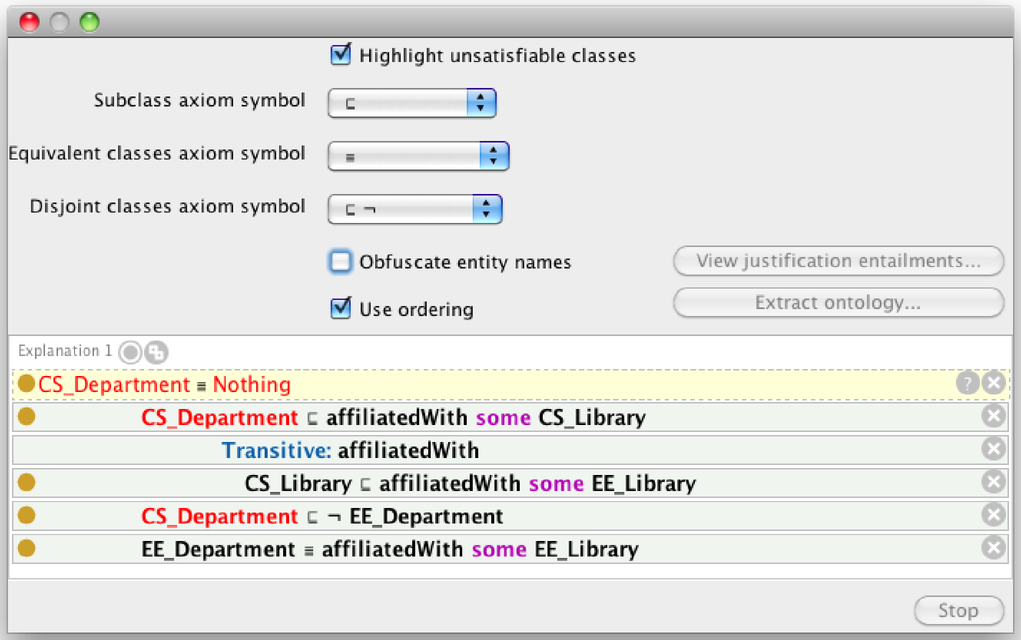
Disjoint classes axiom symbol: ⊑ ¬

☐ Obfuscate entity names

☑ Use ordering

```
● AI_Dept ≡ Nothing
●     AI_Dept ≡ CS_Department and hasResearchArea value AI
●         CS_Department ⊑ affiliatedWith some CS_Library
              Transitive: affiliatedWith
●             CS_Library ⊑ affiliatedWith some EE_Library
●     CS_Department ⊑ ¬ EE_Department
●     EE_Department ≡ affiliatedWith some EE_Library
```

So, the `AI_Dept` is unsatisfiable because its superclass `CS_Department` is, i.e., it is a knock-on effect from `CS_Department`. Does this give sufficient information as to say why `CS_Department` is inconsistent? In fact, it does. See the next screenshot, which is the same as lines 3-7, above.

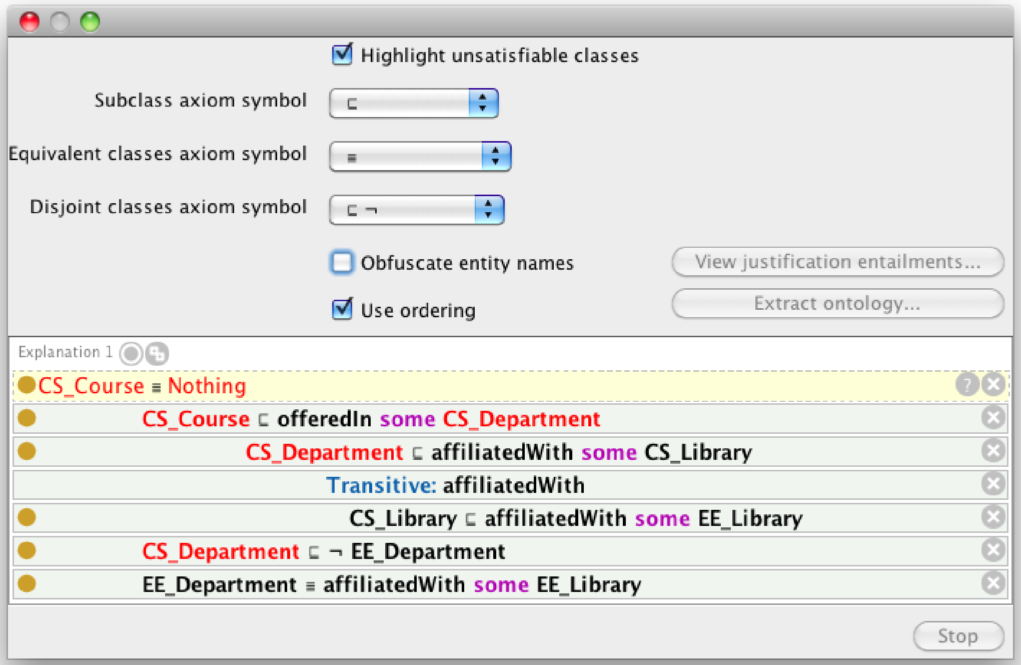


Subclass axiom symbol ⊑
Equivalent classes axiom symbol ≡
Disjoint classes axiom symbol ⊑ ¬

CS_Department ≡ Nothing
CS_Department ⊑ affiliatedWith some CS_Library
Transitive: affiliatedWith
CS_Library ⊑ affiliatedWith some EE_Library
CS_Department ⊑ ¬ EE_Department
EE_Department ≡ affiliatedWith some EE_Library

`CS_Department` is unsatisfiable, because it is `affiliatedWith` some `CS_Library` that, in turn (by transitivity), is `affiliatedWith some` `EE_Library` that belongs to the `EE_Department`, which is disjoint from `CS_Department`. Two 'easy' options to get rid of this problem are to remove the transitivity or to remove the disjointness. Alternatively, we could revisit the domain knowledge; e.g., CS library may not be `affiliatedWith` EE library, but is, `adjacentTo` or disjoint with the EE library.

Let us now consider why `CS_course` is unsatisfiable:


Highlight unsatisfiable classes
Obfuscate entity names
Use ordering
View justification entailments...
Extract ontology...
Stop

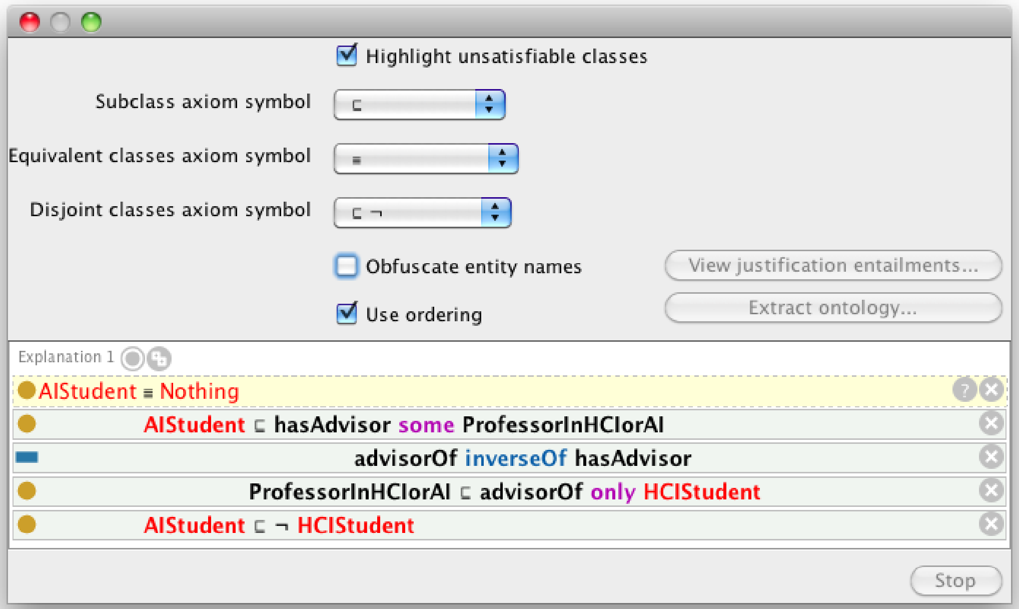
Subclass axiom symbol ⊑
Equivalent classes axiom symbol ≡
Disjoint classes axiom symbol ⊑ ¬

Explanation 1

CS_Course ≡ Nothing
CS_Course ⊑ offeredIn some CS_Department
CS_Department ⊑ affiliatedWith some CS_Library
Transitive: affiliatedWith
CS_Library ⊑ affiliatedWith some EE_Library
CS_Department ⊑ ¬ EE_Department
EE_Department ≡ affiliatedWith some EE_Library

We have again that the real problem is `CS_Department`; fix that one, and `CS_course` is satisfiable, too.

There is a different issue with `AIStudent`. From the explanation in the next screenshot, we can see immediately it has something to do with the inconsistency of `HCIStudent`.

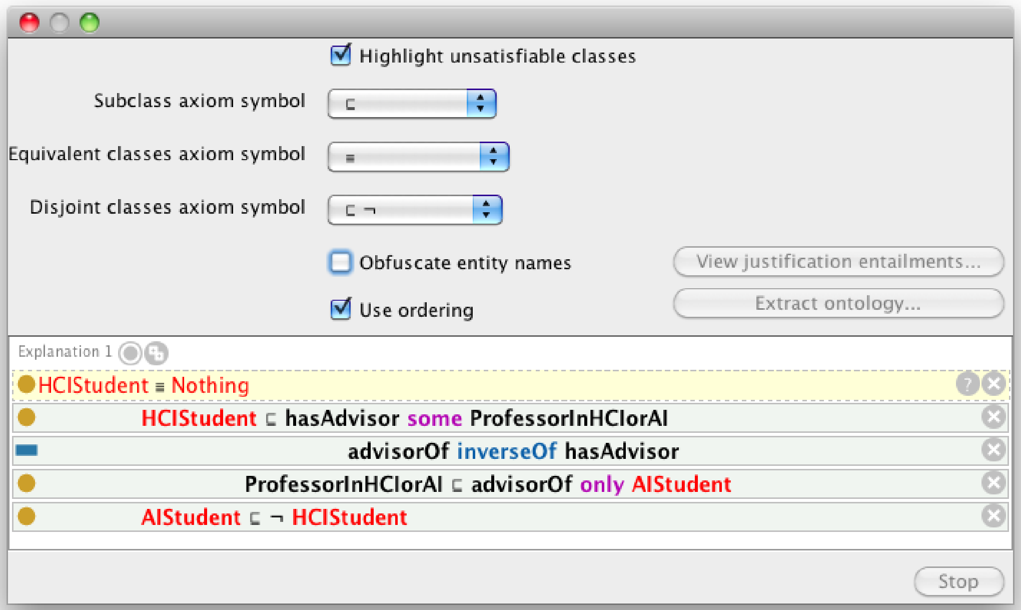

Highlight unsatisfiable classes
Obfuscate entity names
Use ordering
View justification entailments...
Extract ontology...
Stop

Subclass axiom symbol ⊑
Equivalent classes axiom symbol ≡
Disjoint classes axiom symbol ⊑ ¬

Explanation 1

AIStudent ≡ Nothing
AIStudent ⊑ hasAdvisor some ProfessorInHCIorAI
advisorOf inverseOf hasAdvisor
ProfessorInHCIorAI ⊑ advisorOf only HCIStudent
AIStudent ⊑ ¬ HCIStudent

But looking at `HCIStudent` for a clue does not help us further in isolating the problem:



☑ Highlight unsatisfiable classes

Subclass axiom symbol ⊑

Equivalent classes axiom symbol ≡

Disjoint classes axiom symbol ⊑ ¬

☐ Obfuscate entity names

☑ Use ordering

HCIStudent ≡ Nothing

HCIStudent ⊑ hasAdvisor some ProfessorInHCIorAI

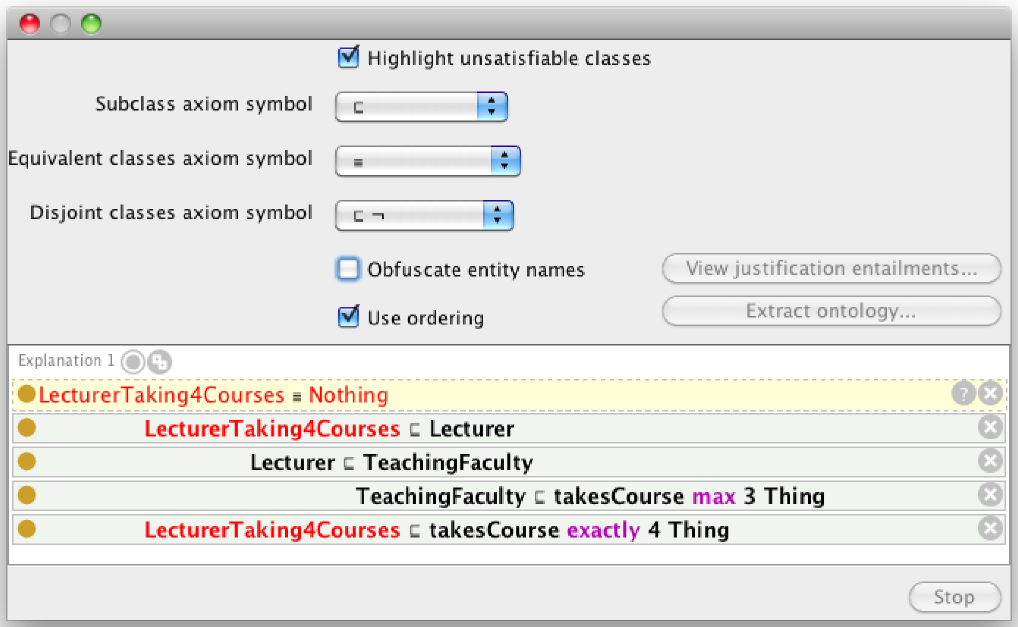
advisorOf inverseOf hasAdvisor

ProfessorInHCIorAI ⊑ advisorOf only AIStudent

AIStudent ⊑ ¬ HCIStudent

Considering the axioms in the explanation only, one can argue that the root of the problem is the disjointness between `AIStudent` and `HCIStudent`, and remove that axiom to fix it. However, does it really make sense to have the union `ProfessorInHCI`*or*`AI`? Not really, and therefore it would be a better fix to change that one into two separate classes, `ProfessorInHCI` and `ProfessorInAI` and have them participating in
`ProfessorInHCI` $\sqsubseteq \forall$`advisorOf.HCIStudent` and
`ProfessorInAI` $\sqsubseteq \forall$`advisorOf.AIStudent`,
respectively.

Last, we have a problem of conflicting cardinalities with `LecturerTaking4Courses`: it is a subclass of `TeachingFaculty`, which is restricted to taking at most 3 courses, which is in conflict with the "exactly 4" of `LecturerTaking4Courses`. This can be fixed by changing the cardinality of either one, or perhaps a lecturer taking 4 courses is not a sub- but a sister-class of `TeachingFaculty`.



☑ Highlight unsatisfiable classes

Subclass axiom symbol ⊑

Equivalent classes axiom symbol ≡

Disjoint classes axiom symbol ⊑ ¬

☐ Obfuscate entity names

☑ Use ordering

Explanation 1

LecturerTaking4Courses ≡ Nothing

LecturerTaking4Courses ⊑ Lecturer

Lecturer ⊑ TeachingFaculty

TeachingFaculty ⊑ takesCourse max 3 Thing

LecturerTaking4Courses ⊑ takesCourse exactly 4 Thing

D.4 Answers Chapter 5

Answer to Exercise 24.
The answer is uploaded on Vula.

Answer to Exercise 25.
There are 39 pitfalls detected and categorised as 'minor', and 4 as 'important'. (explore the other pitfalls to see which ones are minor, important, and critical)

The three "unconnected ontology elements" are used as a way to group things, so are not really unconnected, so that can stay.

ThinAndCripsyBase is detected as a "Merging different concepts in the same class" pitfall. Aside from the typo, one has to inspect the ontology to determine whether it can co with an improvement: what are its sibling, parent and child classes, what its annotation? It is disjoint with DeepPanBase, but there is no other knowledge. It could just as well have been named ThinBase, but the original class was likely not intended as a real merging of classes, at least not like a class called, say, UndergradsAndPostgrads.

Then there are 31 missing annotations. Descriptions can be added to say what a DeepPanBase is, but for the toppings this seems less obvious to add.

The four object properties missing domain and range axioms was a choice by the modellers (see the tutorial) to not 'overcomplicate' the tutorial for novice modellers as they can have 'surprising' deductions, but it would be better to add them where possible.

Last, OOPS detected that the same four properties are missing inverses. This certainly can be added for isIngredientOf and hasIngredient.

D.5 Answers Chapter 6

Answer to Exercise 27.
Some of the differences are: descriptive, possibilism, and multiplicative for DOLCE versus prescriptive and realist, actualism, and reductionist for BFO. You can find more

differences in Table 1 of [Khan and Keet, 2012] and online in the "comparison tables" tab at `http://www.thezfiles.co.za/ROMULUS/`.

Answer to Exercise 28.
There are several differences. The major differences are that DOLCE also has relationships and axioms among the categories using those relationships (i.e., richly formalised), whereas BFO v1 and v1.1 is a 'bare' taxonomy of universals (some work exist on merging it with the RO, but not yet officially). Others are the Abstract branch and the treatment of 'attributes'/quality properties in DOLCE that do not have an equivalent in BFO. The BFO-core has a more comprehensive inclusion of parthood and boundaries than DOLCE.

Answer to Exercise 29.
Informal alignments:

a. dolce:Endurant maps roughly to bfo:Continuant (though actually, more precisely to bfo:IndependentContinuant), dolce:Process as a sub-class of bfo:Process, and dolce:quality to bfo:quality.
b. Amount of Matter, Accomplishment, Agentive Physical Object, and Set do not have a mapping. An example of the possible reasons: Set is abstract, but not existing in nature (hence, by philosophical choice, not in BFO).

A more detailed comparison—or: the results of trying to align DOLCE, BFO, and GFO—is available at `http://www.thezfiles.co.za/ROMULUS/`.

Answer to Exercise 30.
This was discussed in the lecture. The real problem with having a separate relation ontology is that many of the relations don't mean much without the restriction on the domain and range. For instance, take participates in: intuitively we know what it means by understanding the English word, but that won't do for adequate processing by a computer. We can add that an 'object' participates in a 'process'—or: declare the domain of participates in to be 'object' and range to be 'process'—but we have to be precise on what we mean with 'object' and what with 'process', beyond a possibly ambiguous natural language description. The latter can be solved by committing to a foundational ontology, therewith answering questions like: is the 'object' to mean Object in BFO or PhysicalObject in DOLCE or Material_Object in GFO? Or perhaps DOLCE's Endurant, hence, that also non-physical objects and amounts of matter can participate in some 'process'?

Having the relations integrated in the rest of a foundational ontology (cf. separate) definitely increases precision of the representation of the meaning of the relations, and each ontology language has the capability to declare domain and range axioms anyway.

Answer to Exercise 31.
The most often recurring relationships are parthood, participation, constitution, and inherence or dependence.

Answer to Exercise 32.
Options may vary:

1. DOLCE or GFO
2. BFO or GFO
3. Depends on you chosen topic

Answer to Exercise 33.
We will consider ONSET in more detail next week. You can have a look at it w.r.t. the previous question by downloading it from `http://www.meteck.org/files/onset/`

(platform-independent jar file), which can recommend you BFO, DOLCE, GFO, and/or SUMO, depending on the answer you give to one or more of its questions and whether any scaling is applied.

Answer to Exercise 31.
(I use the version with DOLCE in the following answers)

a. To have RockDassie classified as a subclass of Herbivore (still both animals, and physical objects, and physical endurants, and endurants), it needs to have more, or more constrained properties than Herbivore. In Protégé notation, each `Herbivore` is equivalent to:
`(eats only plant) or (eats only (is-part-of some plant))`.
Rockdassies eat grasses and broad-leafed plants. The easiest way to modify the ontology is to add that grasses are plants (already present), that broad-leafed plants are kinds of plants, and that rockdassies eat only grass or broad-leafed plant. This is not to say this is the best thing to do: there are probably also other animals that eat grasses and broad-leafed plants, which now unintentionally will be classified as rockdassies.
b. The ontology does not contain any knowledge on 'living in' and 'nature reserves'. Nature reserves are administrative entities, but also can be considered only by their region-of-space aspect; for the sake of example, let's add $\mathsf{NatureReserve} \sqsubseteq \mathsf{space\text{-}region}$. Trickier is the living, or living in: one could add it as an OWL object property livesIn or as a subclass of Process and add participation relations between that, the nature reserve, and the lions, impalas, and monkeys. The former is less cumbersome, the latter more precise and interoperable (see lecture notes). We'll return to this choice in the bottom-up section.

Answer to Exercise 32.
This sample ontology is available at `http://www.meteck.org/teaching/ontologies/pwEx2.owl`.

No, Human is unsatisfiable.

Reason: $\mathsf{Human} \sqsubseteq \mathsf{ED}$ (EnDurant), it has a property hasPart (to Brain), which has declared as domain PD (PerDurant), but $\mathsf{ED} \sqsubseteq \neg\mathsf{PD}$ (endurant and perdurant are disjoint), hence, Human cannot have any instances.

Answer to Exercise 33.
This is a generalisation of the previous exercise and the example we did during the lecture.

a. The ontologies and their inferences are shown in Table D.1.
b. Thus, there are differences in the deductions. For $\mathcal{O}_1$, the only way to have the ontology consistent is to classify Ed_1 as a subclass of PED, which is possible because PED is a subclass of ED. In the second case, there are several issues (following the reasoner and logic-based explanation): Ed_2 is assumed to be correctly a subclass of AS, but this then runs into problems with $\mathsf{Ed}_1 \sqsubseteq \exists \mathsf{S}.\mathsf{Ed}_2$, because $\mathsf{S} \sqsubseteq \mathsf{R}$ and the range of R is PED and therefore also the range of S is PED (or a subclass thereof), which is disjoint from AS, hence the "$\exists \mathsf{S}.\mathsf{Ed}_2$"-part doesn't work (cannot be instantiated), and therefore Ed_1 cannot be instantiated.
c. We fix the defect by revising the ontology such that the object property hierarchy satisfies the RBox Compatibility service, i.e., the domain and range of S have to be equal or a subclass of the domain and range of R.

Answer to Exercise 34.
The description of the n-ary ODP can be found in the NeON deliverable D2.5.1 on pp67-68. Also, you may wish to inspect the draft ODPs that have been submitted to the

Table D.1: Sample ontologies to illustrate the OWL reasoners and explanation (based on Protégé's explanation feature).

$\mathcal{O}_1$	**OPEs**	**CEs**	**Inferred, with explanation**
	$\mathsf{R} \sqsubseteq \mathsf{PED} \times \mathsf{PED}$ $\mathsf{S} \sqsubseteq \mathsf{ED} \times \mathsf{ED}$ $\mathsf{S} \sqsubseteq \mathsf{R}$	OWLized DOCLE taxonomy, $\mathsf{Ed}_1 \sqsubseteq \mathsf{ED}$, $\mathsf{Ed}_2 \sqsubseteq \mathsf{ED}$, $\mathsf{Ped}_1 \sqsubseteq \mathsf{PED}$, $\mathsf{Ped}_2 \sqsubseteq \mathsf{PED}$, $\mathsf{Ed}_1 \sqsubseteq \exists \mathsf{S}.\mathsf{Ed}_2$, $\mathsf{Ped}_1 \sqsubseteq \exists \mathsf{R}.\mathsf{Ped}_2$	$\mathsf{Ed}_1 \sqsubseteq \mathsf{PED}$: because $D_R = \mathsf{PED}$ and $\mathsf{S} \sqsubseteq \mathsf{R}$
$\mathcal{O}_2$	**OPEs**	**CEs**	**Inferred, with explanation**
	as $\mathcal{O}_1$	as $\mathcal{O}_1$, but with $\mathsf{Ed}_2 \sqsubseteq \mathsf{AS}$ (and $\mathsf{PED} \sqsubseteq \neg\mathsf{AS}$ still holds)	Ed_1 inconsistent: 1. $\mathsf{AS} \sqsubseteq \neg\mathsf{PED}$, 2. $\mathsf{Ed}_1 \sqsubseteq \exists \mathsf{S}.\mathsf{Ed}_2$, 3. $\mathsf{Ed}_2 \sqsubseteq \mathsf{AS}$, 4. $R_R = \mathsf{PED}$ and 5. $\mathsf{S} \sqsubseteq \mathsf{R}$

ODP portal (at `http://www.ontologydesignpatterns.org`, in case you had not found it already).

Answer to Exercise 35.
One could make a Content ODP out of it: for each AssistiveDevice that is added to the ontology, one also has to record the Disability it ameliorates, it requires some Ability to use/operate the device, and performs a certain Function. With that combination, one even can create some sort of an 'input form' for domain experts and administrators, which can then hide all the logic entirely, yet as long as they follow the pattern, the information gets represented as intended.

Another one that may be useful is the Architectural OP: `adolena.owl` now contains some bits and pieces of both DOLCE (endurant, perdurant, and some of their subclasses) and some terms from BFO (realizable), neither of the two ontologies were imported. The architectural ODP can help cleaning this us and structuring it.

D.6 Answers Chapter 7

Answer to Exercise 40.
Phone points conceptual data model to ontology:

a. A sample formalisation is available at `http://www.meteck.org/teaching/ontologies/phonepoints.owl`.
b. Yes, all of it it can be represented.
c. Yes, there are problems. See Figure D.1 for a graphical rendering that MobileCall and Cell are unsatisfiable; verify this with your version of the ontology. Observe that it also deduced that PhonePoint ≡ LandLine.

Answer to Exercise 41.
Integration issues:

a. See Figure D.2
b. Multiple answers are possible due to various design decisions. E.g.,:
   - Did you represent Salary as a class and invented a new object property to relate it to the employees, or used it as a name for an OWL data property (preferably the former)? And when a data property, did you use different data types (preferably not)?
   - Did you add RichEmployee, or, better, Employee that has some property of being rich?

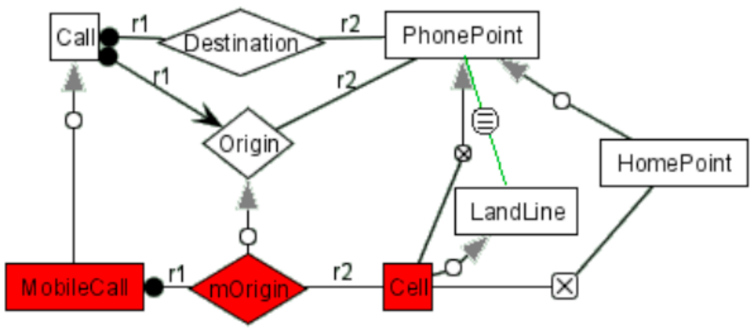


Figure D.1: Answer to 35-b: red: inconsistent class, green: ‘positive’ deduction

- Did you use a foundational ontology, or at least make a distinction between the role and its bearer (Employee and Person, respectively)?

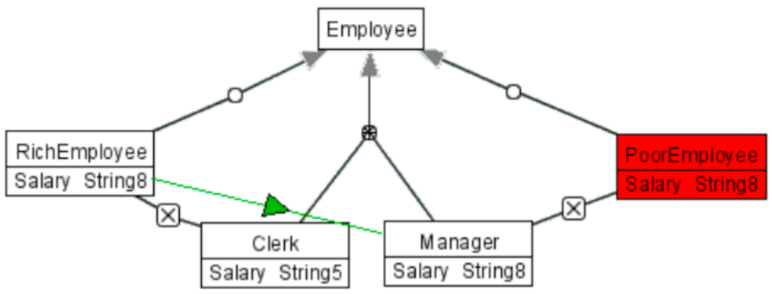


Figure D.2: Answer to 36-a.

Answer to Exercise 42.
Thesauri:

a. Language: SKOS or OWL 2 EL. Why:
   - SKOS: was the purpose of it, to have a simple, but formal, language for ‘smooth transition’ and tagging along with the SW
   - OWL 2 EL: intended for large, simple, type-level ontologies, and then still some reasoning possible

b. Regarding mass media, films and news media: not necessarily, but to be certain, check yourself what the definition of `Mass Media` is, when something can be called `News Media`, and then assess the differences in their properties.
   `Propaganda` has as broader term `Information Dissemination`, but a characteristic of propaganda is dissemination of *mis*information.

Answer to Exercise 43.
Principally:

- expressive foundational ontology, such as DOLCE or GFO for improved ontology quality and interoperability
- bottom-up onto development from the thesaurus to OWL
- integration/import of existing bio-ontologies
- Domain ontology in OWL taking the classification of the chemicals, define domain & range and, ideally, defined concepts
- Add the instance data (the representation of the chemicals in stock) in the OWL ABox (there are only 100, so no real performance issues), and add a dummy class disjoint from DruTopiate destined for the ‘wrong’ chemicals

- Take some suitable reasoner for OWL 2 DL (either the 'standard' reasoners or names like fact++)
- Then classify the instances availing of the available reasoning services (run fact++ etc.): those chemical classified as instances of the 'ideal chemical' are the candidate for the lab experiments for the drug to treat blood infections.
- Alternatively: add DruTopiate as class, add the other chemicals as classes, and any classes subsumed by DruTopiate are the more likely chemicals, it's parents the less likely chemicals.
- Methods and methodologies that can be used: single, controlled ontology development, so something like METHONTOLOGY will do, and for the micro-level development something like OD101, ontospec, or DiDON.