# Ontology Representation : design patterns and ontologies that make sense

Hoekstra, R.J.

**Publication date**
2009
**Document Version**
Final published version

# Ontology Representation

Design Patterns and Ontologies that Make Sense

Rinke Hoekstra

SIKS Dissertation Series No. 2009-15

The research reported in this thesis has been carried out under the auspices of SIKS, the Dutch Research School for Information and Knowledge Systems.

# ONTOLOGY REPRESENTATION

## Design Patterns and Ontologies that Make Sense

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof. dr. D.C. van den Boom
ten overstaan van een door het college voor promoties
ingestelde commissie,
in het openbaar te verdedigen in de Agnietenkapel
op vrijdag 18 september 2009, te 12:00 uur

door

Rinke Jan Hoekstra

geboren te Zaanstad

Promotiecommissie

| | |
|---|---|
| Promotor: | prof. dr. J.A.P.J. Breuker |
| Copromotores: | prof. dr. T.M. van Engers |
| | dr. R.G.F. Winkels |
| Overige leden: | prof. dr. F. van Harmelen |
| | prof. dr. E. Motta |
| | prof. dr. F.J.M.M. Veltman |
| | prof. dr. B.J. Wielinga |
| | dr. B. Bredeweg |

Faculteit der Rechtsgeleerdheid

*To Eefje and Lieve*

# Contents

# Chapter 1

# Introduction

"I pretty much try to stay in a constant state of confusion just because of the expression it leaves on my face."

*Johnny Depp*

## 1.1 Introduction

An intelligent computer system needs to maintain an internal representation of that part of reality it is to be intelligent about: it needs to understand its domain. This understanding is commonly captured in models on the basis of which a computer can perform automated reasoning. For instance, a suitable model of 'table' will allow a machine to infer that each occurrence of a table has four legs. Models are often not a direct reflection of the domain itself, but rather represent our knowledge of that domain. The construction of models thus involves a mapping – a translation – between human understanding and that of a computer. In fact, the translation of human knowledge to computer models is one of the most profound problems in Artificial Intelligence (Newell, 1982).

Over the course of several years, I have worked on many projects that involved the design and construction of such models, for a variety of purposes. The specification of an ontology for a large insurance company in the CLIME project (Boer et al., 2001; Winkels et al., 2002), made me aware that not only this modelling is a lot of work, but also that mistakes are easily made (Hoekstra, 2001).[1] The systematic representation of the contents of regulations in the E-POWER project (van Engers et al., 2000), illustrated not just the benefits of this approach, but simultaneously highlighted the enormous effort and scru-

[1] CLIME, Cooperative Legal Information Management and Explanation, Esprit Project 25414.

tiny required from the people that actually build the models.[2] Furthermore, both the E-POWER and KDE projects (Jansweijer et al., 2001) made it acutely clear how important the language is in which the model is expressed.[3] Coming from academia, it was puzzling to see how easily one defaults to inexpressive but intuitive languages (or even just database tables), rather than well-wrought languages that support reasoning. What is more, it was very difficult to defend a more principled approach as available languages did not have widespread tool support (yet), nor did they offer even the trivial reasoning capabilities our users *were* interested in: then what makes them 'better' languages? Was I a puritan?

When I first set out to work on this book the objective was to investigate a formal representation of physical causation for the purposes of automatic attribution of liability in legal cases. It was the natural follow-up on the work of Lehmann (2003), who gave an overview of the different forms of causation involved in liability and responsibility attribution. Our approach, described in Breuker and Hoekstra (2004b); Hoekstra and Breuker (2007), was based on the hypothesis that legal causation (which is a form of liability) can only be established given full knowledge of the chain of events leading from some initial event (an action) to an undesirable state (some damage).

This approach turned out to be problematic for two reasons. First of all, the subject of causation – and legal causation and liability in particular – plays a prominent role in philosophy and legal theory. One cannot present a formal, computational representation of causation without taking a stance in this discussion. Perhaps this is because a necessary step for automated reasoning, namely the construction of a *computational model* of a theory, may be confused with the presentation of a *formal theory*. As no established theory on causation directly fit our needs, there was no escaping: even though we did not purport to present a new theory on causation, our model was considered as such.

The second problem was more practical: the representation of even a very simplistic model of physical causation turned out to be quite difficult (though not entirely impossible, see Section 7.4). The heart of the problem was our requirement that the existence of a causal relation was to be inferred on the basis of a description of multiple successive situations and the changes between them, rather than only from other causal relations. An event (and consequently causes) is then classified by recognising the two situations between which it occurs. First we used the Web Ontology Language (OWL, Bechhofer et al. (2004)), a language optimised for classification, but its expressiveness was too limited to enforce relations between the two situations. For instance, it is impossible to express that a change occurs to two states of a *single* object instead of to two different ones (Hoekstra et al., 2006). We were able to express these restrictions using the Prolog language; but it required the implementation of a custom classifier. Clearly, that was not the solution either.

The ESTRELLA project brought the development of a legal knowledge interchange language (LKIF, Boer et al. (2007a)) and a core ontology for the legal domain (LKIF Core, see Hoekstra et al. (2008) and Chapter 6).[4] An important

[2] E-POWER, European Program for an Ontology based Working Environment for Regulations and Legislation, IST-2000-28125.

[3] KDE, Knowledge Desktop Environment, Esprit Project 28678.

[4] ESTRELLA, European project for Standardized Transparent Representations in order to Extend Legal Accessibility, IST-2004-027655.

role of this ontology is to provide the conceptual basis for more expressive and elaborate models that can be used to provide complex reasoning services. In practice, this meant a revisiting of the problems I faced in my work on causation. For, how to combine representations in OWL with highly expressive languages that have a direct correspondence with legal theory? Secondly, to allow for intelligent reasoning, the OWL definitions in the ontology had to be extended to more intricate descriptions: the language was pushed to its limits.[5]

## 1.2 Questions

These experiences expose several interesting questions pertaining to the representation of human knowledge in a computer model. I briefly discuss the five most prominent ones here:

**How can the quality of models be ensured?** Evaluating the quality of a model depends on the criteria used in its evaluation. What quality criteria and requirements apply to representations of knowledge, and how do they interact? To what extent do design principles, methods, and the choice of language contribute to the quality of our models?

**Can the design of models be facilitated, or made easier?** Building a formal computational model is both difficult and a lot of work, while the possible pay-off is not always immediately clear. What solutions have been proposed to lower this threshold, and how do they perform in practice?

**To what extent do theory and practice go hand in hand?** Formal theories can be said to reflect a profound understanding of a domain; but are they adequate computer models? To what extent can and should criteria that hold for formal theories be applied to models designed to be used in an intelligent system?

**What is the rationale behind representation languages?** The field of artificial intelligence boasts a large number of languages that can be used to construct models. Although these languages can be very different, each has been designed with a specific purpose in mind. The question is, how does one know which language is appropriate for the purpose at hand?

**How do limitations in expressiveness affect models of a concrete domain?** Every formal language distinguishes itself by offering a different set of primitives that can be used to construct models. The choice for a language is therefore a commitment to the limitations of that set of primitives. What does this commitment mean in practice, for a concrete domain?

In this book I report on my quest to find answers to each of these questions. Instead of treating each question in turn, they are rather used as background against which the following chapters unfold. Chapter 2 presents the base line

[5]For examples of some of the problems we faced, see Breuker et al. (2007); Hoekstra (2008); Klarman et al. (2008); van de Ven et al. (2008b); Hoekstra and Breuker (2008); Hoekstra et al. (2008), and Chapter 7.

for answering all five questions. Chapter 5 has a strong focus on the first two questions, while Chapter 4 is primarily concerned with the issue raised in the third question. Chapter 3 elaborates on the second chapter to improve a better understanding of the fourth and fifth question. Chapters 6 and 7 present the consequences of the discussion in the preceding chapters for the case of a concrete domain. Chapter 6 emphasises the first and third question, and Chapter 7 focuses on the second, fourth and last question.

## 1.3 Ontologies

The following chapters discuss the questions introduced in the previous section in the context of a particular type of computer model: ontologies. This section explains what makes the design of ontologies such a suitable domain for this investigation.

'Ontology' is a term that people who have come across the subject of the Semantic Web will be familiar with: the two go hand in hand. The use of ontologies is widespread; their utility is universally acknowledged and they are the talk of the town at many conferences. However, it is quite hard to find out what exactly ontologies *are*, and why they play such a prominent role. The term 'ontology' is clearly overloaded, bringing together insights from philosophy, artificial intelligence, systems engineering, information management, computational linguistics, and cognitive psychology. The cacophony of voices resulting from this interdisciplinary interest leads to heated and interesting debates but can be quite bewildering to the ingenuous newcomer who simply wants to *use* the technology.

In answering their principal question – what *is* an ontology – experts are implicitly biased with respect to their own perspective. As I will discuss, the interplay between abstract, theoretical considerations and practical requirements render it impossible provide a single correct answer. This book attempts to elucidate the different perspectives, and emphasises one interpretation, that of *knowledge representation*. Knowledge representation is a field of artificial intelligence that tries to deal with the problems surrounding the design and use of formal languages suitable for capturing human knowledge. The ultimate goal of this formal representation is to enable intelligent automated reasoning on the basis of that knowledge. This *knowledge-based reasoning* takes place within systems that are designed, as a whole, to perform tasks which are normally carried out by human experts. These tasks typically require the consideration of large amounts of data, e.g. where human reasoning is error prone, or just tedious. Analogous to software engineering, knowledge *engineering* is the field that concerns itself with the specification and design of such systems. It is an important aspect of *knowledge acquisition*, the general problem of how to extract and organise knowledge from human experts in such a way that it is implementable in a reusable manner.

The design of ontologies plays a prominent role in both knowledge representation and engineering. The field of ontology engineering has brought forth numerous methodologies and design principles on the subject of ontology construction. The Web Ontology Language (OWL) is a prominent member of the knowledge representation languages family, designed specifically for the representation of ontologies. Its expressiveness is restricted to guarantee favour-

able computational properties. Around the start of the ESTRELLA project, the OWLED community was soliciting support for a new working group at the W3C – the internet's main standardisation body.[6] The working group was to follow-up on a member submission by several members of the community that proposed a number of extensions to OWL: an opportune moment for extending and exploring the expressiveness bounds of this language.

The specification of ontologies in OWL is often considered difficult. This drives tool development, e.g. Protégé 4 and its plugin library,[7] explanation facilities, a continuous refinement of methodologies, and (more recently) the specification of ontology design patterns. Furthermore, a fair number of ontologies have been developed that are targeted to provide a (generic) unifying framework for multiple domains. It is generally held that such ontologies aid the construction and reusability of more specific domain ontologies and knowledge representations.

In short, the construction of ontologies is a well-established topic of research that provides ample inspiration for answering the questions iterated in the previous section:

- The role and quality of ontologies have been topics of research for quite some time. An assessment of the state of the art in the context of questions one and two provides insight as to what extent these questions are (or can be) answered, and what issues should be considered.
- The term 'ontology' originates in philosophical theory, but is adopted by the more application-oriented field of artificial intelligence and the Semantic Web. The interplay between these fields is an enticing use case for investigating the third question.
- Ontologies can be expressed using a tailor-made knowledge representation language that is subject to several important limitations. The characteristics of this language shed light on the requirements imposed on the development of knowledge representation languages, and thus on the fourth question.
- Several ontologies have been put forward that can be regarded as gold standard for ontology development. These prominent examples do not just illustrate some perspective on the quality and design of ontologies, but contribute to insight in the trade-off between theory and practice (question three), and ontology specification using a particular language (questions four and five).
- Ontologies play an important role on the Semantic Web, and are widely used by a very diverse group of people. In other words, they are not just abstract, theoretical notions that do not affect practice, but have a significant user base that will benefit greatly from tangible guidelines that would result from an answer to all five questions. In particular, a worked-out example of the simultaneous application of these insights to a concrete

[6] W3C, World Wide Web Consortium, `http://www.w3.org`. OWLED, OWL Experiences and Directions, see `http://www.webont.org/owled/`.

[7] Protégé 4 is an OWL ontology editor developed by the universities of Stanford and Manchester. See `http://protege.stanford.edu`.

domain (question five) may provide better understanding of the issues involved than a separate consideration would.

## 1.4 Contribution and Overview

The following chapters explore the topics introduced in this chapter as follows:

**Chapter 2 – Knowledge Representation** gives a historical overview of the field of knowledge representation and acquisition. It discusses the quest for a knowledge representation language that has a clearly understood status with respect to the knowledge that it can represent. Important in this light are issues of maintenance and reusability of knowledge based systems. These requirements show that a knowledge representation language is not 'just' a generic formal language. This chapter presents arguments for a language that has well-defined computational properties and can be used to build task independent knowledge system components.

**Chapter 3 – Semantic Web** introduces the ideas underlying the Semantic Web, the Web Ontology Language, and in particular its successor OWL 2. Both highly expressive web-based knowledge representation languages. The chapter shows how the requirements formulated in Chapter 2 interact with the open nature of the web, and explains the rationale and limitations of the primitives available in OWL 2. This language is selected as base line for the discussion of ontologies, methodologies and design patterns in the subsequent chapters.

**Chapter 4 – Ontologies** discusses the widely varying conceptions of what (an) ontology *is*, paying attention mainly to its use in philosophy and in knowledge representation. This discussion makes clear that a lot of the confusion surrounding ontologies stems from an obfuscation of the two perspectives, and proposes a more crisp distinction between different types of ontologies. The role of ontologies that are expressed using knowledge representation languages is explained and adopted as central to the task of ontology engineering discussed in the subsequent chapters.

**Chapter 5 – Ontology Engineering** presents an overview of methodological approaches to building ontologies. It highlights several design principles for ontology construction. In particular, the role of ontologies as reusable knowledge components leads to a number of restrictions both with respect to what an ontology contains, and as to how it may be reused. For each of these topics, this chapter discusses whether and how these can be reconciled with the knowledge representation perspective on ontologies described in Chapter 4. Furthermore, the chapter proposes a refinement of current views on reuse, design patterns, and of the kind of knowledge expressed in ontologies.

**Chapter 6 – Commonsense Ontology** applies the insights of the preceding chapters to the construction of a core ontology for the legal domain: LKIF Core. This ontology is designed to support the reasoning task of a knowledge based

system; it is specified in OWL 2, and compared to a number of existing upper and core ontologies. An important difference with these ontologies is that it is based on insights from cognitive science rather than philosophy, and reflects a common sense, rather than theoretical perspective. The discussion of the LKIF Core ontology serves to illustrate the consequences of the considerations introduced in the preceding chapters as applied to a concrete domain; both considering the knowledge representation language used, and the stance with respect to (legal) theory.

**Chapter 7 – Design Patterns** describes a suite of design patterns that have been implemented in the LKIF Core ontology: a *diamond* shaped pattern is applied to the definition of transactions; *summarisation* of reified relations is used to define social concepts; and *sequences* are employed to define processes and causal relations. These patterns are combined in the definition of actions.

Where the preceding chapter provides a high level description of the perspective and design decisions underlying the LKIF Core ontology, this chapter zooms in to the level of the OWL 2 knowledge representation language to illustrate how these considerations are applied in the definition of concepts central to both legal and other domains. At this level, the conceptual insights of the preceding chapters are directly confronted with expressiveness bounds of OWL 2. The discussion of design patterns explicitly addresses the trade offs involved in their specification, and explains useful strategies for extending and combining them in more elaborate structures. The way in which these patterns are presented is meant to maximise insight in the task of ontology design as a whole.

# Chapter 2

# Knowledge Representation

> "Once the characteristic numbers of most notions are determined, the human race will have a new kind of tool, a tool that will increase the power of the mind much more than optical lenses helped our eyes, a tool that will be as far superior to microscopes or telescopes as reason is to vision."
>
> *Gottfried Wilhelm Leibniz, Philosophical Essays*

## 2.1 Introduction

The goal of AI research is the simulation or approximation of human intelligence by computers. To a large extent this comes down to the development of computational reasoning services that allow machines to solve problems. Robots are the stereotypical example: imagine what a robot needs to know before it is able to interact with the world the way we do? It needs to have a highly accurate internal representation of reality. It needs to turn perception into action, *know* how to reach its goals, what objects it can use to its advantage, what kinds of objects exist, etc. Because this problem solving takes place in a different environment (inside a computer) than human problem solving, it is subject to different restrictions, such as memory capacity, processing power and symbol manipulation. Where human reasoning can resort to a wide array of highly redundant patterns, machines will inevitably resort to parsimonious and incomplete representation, suitable only for solving a particular set of problems.

The field of knowledge representation (KR) tries to deal with the problems surrounding the incorporation of some body of knowledge (in whatever form) in a computer system, for the purpose of automated, intelligent reasoning. In this sense, knowledge representation is *the* basic research topic in AI. Any artificial intelligence is dependent on knowledge, and thus on a representation of that knowledge in a specific form.

The history of knowledge representation has been nothing less than turbulent. The roller coaster of promise of the 50'ies and 60'ies, the heated debates of the 70's, the decline and realism of the 80's and the ontology and knowledge management hype of the 90's each left a clear mark on contemporary knowledge representation technology and its application. In particular, the idea of a Semantic Web (discussed in Chapter 3) led to the integration of insights from two distinct fields in symbolic AI: knowledge *representation*, and knowledge *acquisition* (expert systems). Two areas that had showed little or no interaction for three decades, at least not significantly, since the divide between *epistemic* and *heuristic* aspects of an intelligent system (McCarthy and Hayes, 1969).

In this chapter I give an overview of the historical origins and rationale of knowledge representation, and the family of languages known as *description logics* in particular. These languages turned out to play an important role in the development of semantics on the web, discussed in Chapter 3, and could not have reached their current prominence without the wide adoption of the word 'ontology' in the AI literature of the nineties (see Chapter 4).

## 2.2 Two Schools

In the early second half of the 20th century, AI housed two very different schools of thought. The first was very much based on the idea that knowledge is best captured using a general purpose, clean and uniform language: *logic*. With roots in philosophy, it was oriented towards the adequate representation of our theoretical understanding of the *structure* of the world, and assumed that a small set of elegant first principles can account for intelligence. The second school's main interest was the approximation of *human* intelligence, and human *behaviour* in particular. Its main proponents had a background in psychology and linguistics, rather than philosophy or mathematics, and were less concerned with rigourous formal semantics. They built systems that 'just work', based on the assumption that human intelligence is a hodgepodge of many different ad hoc conceptions and strategies. Roger Schank coined the two groups the *neats* and the *scruffies*, respectively.

In short, research in KR can be roughly categorised as having either a *philosophical* or *psychological* nature. In an influential paper McCarthy and Hayes (1969) discussed a number of fundamental issues for AI (amongst which the famous frame problem). They operationalise (artificial) intelligence as follows:

> "...an entity is intelligent if it has an adequate model of the world (including the intellectual world of mathematics, understanding of its own goals and other mental processes), if it is clever enough to answer a wide variety of questions on the basis of this model, if it can get additional information from the external world when required, and can perform such tasks in the external world as its goals demand and its physical abilities permit."
>
> (McCarthy and Hayes, 1969, p.4)

This definition introduces the distinction between a *representation* of the world, and a *mechanism* that uses problems and information expressed in that representation to perform problem solving and decision making. Artificial

| | **Semantics** (Epistemological Adequacy) | **Reasoning** (Heuristic Adequacy) |
|---|---|---|
| **Philosophy** | First Order Logic | Theorem Provers (exhaustive, combinatorial) |
| **Psychology** | Semantic Nets, Frames, (Rules) | Problem Solvers (goal directed, rational) |
| **Practice** | Rules | Expert Systems |

Table 2.1: Schools and systems in Knowledge Representation

intelligence systems should attain a balance between both *epistemological* adequacy and *heuristic* adequacy.

The distinction between these approaches was very much evident in AI research in the seventies. Mylopoulos (1981) organised the schools in a taxonomy; KR languages can first of all be divided into procedural and declarative ones. The first is aimed at attaining heuristic adequacy, the second has an epistemological perspective. The declarative languages can be further subdivided into logic-based and semantic network ones, see Table 2.1. It wasn't until the end of the seventies that so-called *hybrid* systems attempted to combine declarative and procedural views (see Figure 2.1).

### 2.2.1 Intelligence by First Principles

The idea of automated intelligent reasoning has been around for a long time, and can be traced back to ancient Greece. Aristotle's *syllogisms* are often seen as the first example of a formalisation of valid reasoning. With the separation of mind and body in the 17th century by Descartes, and in common sense, the road was opened up to apply newly found mathematical insights to model and imitate parts of human thought as mechanistic processes. Perhaps the most appealing examples are Pascal's arithmetic machine for addition and subtraction, and Leibniz' improvements to it to support multiplication, division and computing the square root.

In the mean time other great minds, such as John Wilkins (the first secretary of the Royal Society) were busy working on a systematic account of all of human knowledge using his *Real Character*. The real character encoded words in such a way that each had a unique non-arbitrary name. For this, Wilkins used a three-layered tree structure. All concepts are distributed over forty Genuses; these in turn are divided into Differences which are separated as Species. Each of these layers adds one or more letters, such that any path through the tree can be represented as a unique four-letter word.

More influential, however was Leibniz' invention of the binary system of numbers that lies at the heart of his calculator. He entertained the thought of encoding 'notions' as unique binary encoded numbers. Using a simple method

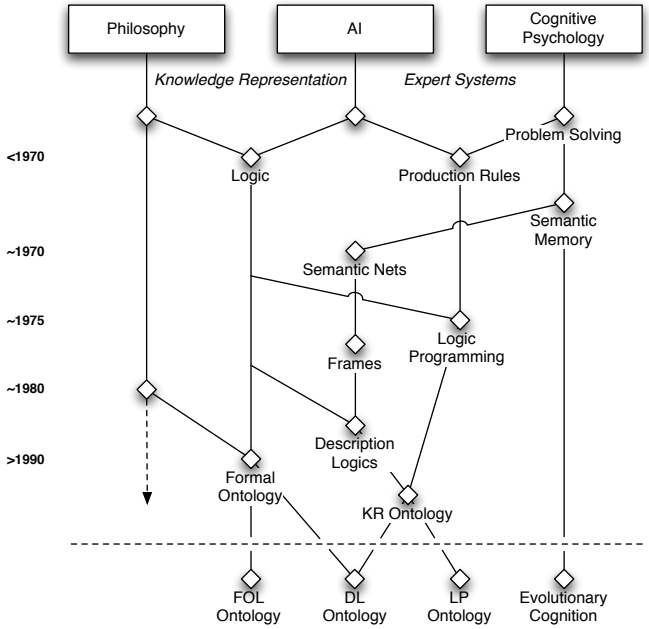


Figure 2.1: History of knowledge representation

of combining these numbers by applying mathematical operators, a machine – the *calculus ratiocinator* – could be built that would 'think'. Unfortunately, gathering all notions proved to be too formidable a task, and for various reasons – his work on the differential calculus beckoning – Leibniz was unable to continue his work in this direction. A few decades later Linnaues was more successful when he built his *Systema Naturae* of a more limited domain.

In Leibniz' view, intelligent reasoning is a form of *calculation*: the manipulation of symbols according to a set of logical axioms. Still (or even more) prevalent once computers became available, logic based knowledge representation was very popular during the 1960's after the development of automatic theorem proving using *resolution*. The idea of a general-purpose logical engine fuelled the idea that logics could be used as the basis of all intelligent action. Despite the fact that logic based knowledge representation has the advantage of a well-understood formal semantics, standard inference rules, and relatively simple notation, it has several drawbacks as well (Mylopoulos, 1981).

Logic based languages did not provide a way to organise knowledge in separately understandable modules. And as time progressed it became clear that this engine was not "powerful enough to prove theorems that are hard on a human scale", which lead to the "great theorem-proving controversy of the late sixties and early seventies" (Newell, 1982, p.90-91). Instead of being universally applicable, research in theorem proving increasingly focussed on smaller, theoretically hard topics. The result was a rather allergic reaction to anything smelling of *uniform procedures*, and at the start of the seventies it seemed that logic as knowledge representation language was very much done for in AI Hayes (1977); Newell (1982).

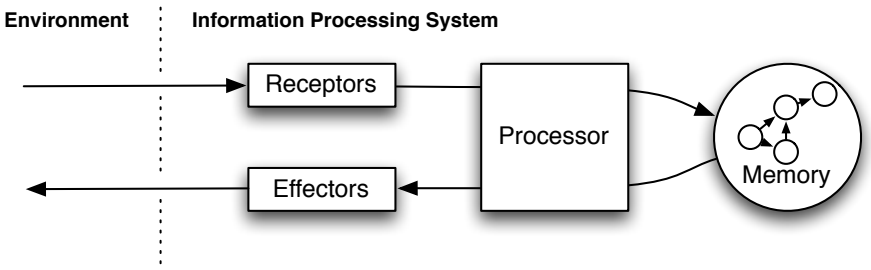


Figure 2.2: Structure of an Information Processing System (adapted from Newell and Simon (1972))

Another problem was that logics are not well suited to represent procedural knowledge. The PROLOG programming language (Kowalski, 1974; Bratko, 1986) was designed to alleviate this problem by the interpretation of implications as procedure declarations (see Figure 2.1).

### 2.2.2 Production Systems

At the start of the seventies, much of AI research was targeted at building psychologically sound computer models. Such models were used in cognitive psychology research for the study of both language semantics and human reasoning. It soon became evident that people within the two areas of research entertained differing views on what 'knowledge' is. While one group maintained that knowledge is all about 'how' (the heuristic view), the other group advocated the 'what' (the epistemological view). The latter is discussed in Section 2.2.3.

During the 1950s, Newell and Simon (1972) developed the *Logic Theorist* and the *General Problem Solver* (GPS) programs which were able to perform various tasks using a combination of theorem proving and heuristics. Although they started out in a similar vain as the purist logicians, they soon developed a quite different approach. In their view, human thinking and problem solving is by information processing: "the human operates as an information processing machine"(Newell and Simon, 1972, p.21). Even though this *information processing system* (IPS) perspective soon turned out to be overly simplistic as a correct model of the human mind; it is a useful abstraction, and has developed into a major knowledge representation paradigm. The general structure of an IPS is that of a processor that interacts with its environment – using a receptor and effector – and stores information about the environment in its memory (see Figure 2.2).

The processor consist of a set of elementary information processes (eip) and an interpreter that determines the sequence of processes to be executed as a function of the symbols stored in memory. This view was very much oriented towards the way in which computers can be used to do *thinking*. Of primary concern, here, were the questions as to 1) how elementary information processes should be expressed, and 2) what strategies should be implemented as part of the interpreter.

Systems that follow the IPS architecture of Newell and Simon are generally a type of *production* system (Buchanan and Shortliffe, 1984). These systems were first conceived of as a general computational mechanism by Post (1943), used to describe the manipulation of logical symbols. In its simplest form, a production rule consists of a left hand side (condition) and a right hand side (action), usually in the form of an `if ...then ...` statement. A production rule is essentially the operationalisation of a reasoning step (i.e. an eip) in an IPS: given some input structure of symbols, the rule produces a new (modified) structure of symbols. An interpreter iteratively evaluates all productions in the system until it finds one that matches one or more symbols stored in memory. This evaluation of the condition of rules is a passive operation that has no impact on those symbols. When the interpreter evaluates some input to the conditions of a rule, it is said to 'fire', and performs the operations specified on the right hand side on relevant symbols in memory. Because of its dependency on the order in which the interpreter carries out evaluation, a production is not applied in the same way as the full combinatorics of logical implication. Where the consequent of an implication necessarily holds at all times – all information implicit in the knowledge base holds at all times – the consequent of a production rule only comes into effect after the rule has fired.

Production rules were (and still are) used for a wide variety of applications. They can be categorised according to two views: as a means for psychological modelling on the one hand (as in IPS), and for *expert systems* on the other. In cognitive psychology, production rule systems were part of an effort to create programs that capture human performance of simple tasks. This performance includes typical human treats such as forgetting, mistakes etc. and rules were a promising paradigm for capturing heuristics in human problem solving. For these scruffies, human intelligence was rather a "particular variety of human behaviour" (Davis et al., 1993, p.10); the 'intelligence' of reasoning can only be assessed by virtue of its correspondence to human intelligence, and not necessarily by whether it is clean and logical. To them, production systems provided a clear formal way to represent the basic symbol processing acts of information processing psychology (Davis and King, 1984). The production rule semantics allowed an escape from the nothing-or-all inferences of theorem proving, and could be used to capture the local, rational control of problem solving.

During the eighties, rule-based knowledge representation was applied in a number of large scale projects, and found its way into many enterprise industrial and government applications. Because of their rather practical, application oriented perspective, focus shifted from a cognitive perspective to building large knowledge-based systems, and creating and maintaining elaborate models that capture expert knowledge. Knowledge-based expert systems, emphasise problem-solving *performance* at the cost of psychological plausibility. Production rules are used to capture expert knowledge about a particular task or domain, and enable the system to support, augment or even surpass human problem solving. Production rules can be modified and extended relatively easily, which makes them a convenient paradigm for incremental system development. A well known example of such a system is is the MYCIN expert system for medical diagnosis (Buchanan and Shortliffe, 1984). Rather than attempting to simulate diagnosis by human experts, it captures and formalises the (often implicit) knowledge, i.e. the 'rules of thumb' used by those experts, into the form of production rules. Because of the emphasis on performance,

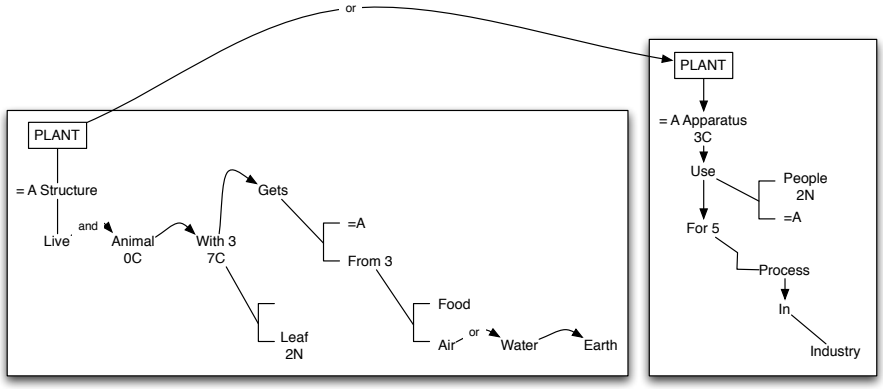


Figure 2.3: A semantic network, using the notation of Quillian (1966)

interest soon grew towards the improvement of the interpreter with more efficient strategies.

### 2.2.3 Semantic Networks

While production systems were rather good at mimicking the heuristics of human problem solving, they were severely limited when it comes to another major area in AI: natural language understanding. Looking at it in terms of an IPS, a natural language processing system is all about mapping terms and structures in natural language to their cognitive interpretation in memory. In other words, lexical terms are to be *grounded* in a model that represents their semantics, where the semantics should mimic the human understanding of these terms in memory. This application area brought forth a number of very influential knowledge representation technologies that count as the direct predecessors of the languages currently used for representing knowledge on the Semantic Web.

The theory of *semantic memory* by Quillian (1966), is based on the idea that memory consists of associations between mental entities, i.e. semantic memory is *associative* memory (Anderson and Bower, 1973). Quillian's semantic memory can be depicted using *semantic networks*, directed graphs where nodes are terms and the relationships between those terms are represented as arcs (see Figure 2.3). In semantic networks, different senses of a word concept (a node) can be organised into *planes*, and can be related through *pointers* (edges). Pointers within a plane form the structure of a definition. Pointers leading outside a plane indicate other planes in which the referenced words themselves are defined. The use of planes and pointers allowed the 'import' of a word definition by reference. Quillian (1966) distinguished five kinds of pointers: subclass, modification, disjunction, conjunction, subject/object.

Though graph-based representations had been used extensively in the past for a wide variety of representations, Quillian was the first to use semantic networks for representing human knowledge:

> "His intent was to capture in a formal representation the 'objective' part of the meanings of words so that 'humanlike use of those meanings' would become possible"
>
> (Brachman, 1979, p. 5)

Underpinning the cognitive perspective of this approach, was his later research using reaction time in assessing the factual truth of statements (Collins and Quillian, 1969, semantic verification) to test the psychological plausibility of semantic network models. Of particular interest was the question as to whether the retrieval of inferred property values (over the subclass relation) would take more time than directly represented property values. The fact that indeed this was the case provided backing for *property inheritance* over a superclass-subclass taxonomic hierarchy in semantic network representations of human semantic memory. Furthermore, he intended his model to be suitable for automatic *inference*, allowing for querying information implicit in the model. Effectively turning the semantic network paradigm into not only a representation but a *simulation* of human memory.

The expressive power of the original semantic networks soon became too restricted, and a number of innovations and extensions followed. Quillian's original set of five link types turned out to be insufficient, and was superseded by the ability to type pointers using *named* attributes, i.e. a means to use a token to point to a type. Furthermore, a distinction was introduced between concepts and examples (later *instances*), in Carbonell (1970). These innovations led to a plethora of widely variant, rather unprincipled, semantic network 'languages'. Many of which applied the technique to domains other than psychology.

A true show-stopper, however, was that new link types, and even concept types, were not explained and the interpretation of semantic networks was left to the 'intuition' of the reader (Brachman, 1979): semantic networks did not really have semantics. For instance, both the concept–instance distinction and the type–token distinction were obfuscated by the use of the infamous 'IS-A' link (Woods, 1975; Brachman, 1983).

Perhaps most manifest to current readers was the critique that the networks made no distinction between domain level constructs – conceptual relations in the domain – and knowledge structuring principles such as the subclass relation. As semantic networks relied heavily on graphical notation, this is most apparent in the uniformity of presentations. Woods stressed the importance of considering the semantics of the representation *itself*.

To summarise, semantic networks were developed as part of an effort in psychology to represent human semantic memory. Although they have been successful in many ways, they suffered from lack of proper semantics: "The 'semanticness' of semantic nets lies in their being used in attempts to represent the semantics of English words." (Brachman, 1979, p. 26).

### 2.2.4 Frames

The semantic network model received criticism from cognitive science itself as well. Most notable in this respect is Minsky (1975),[1] who argued against

[1] Though, as is common his ideas had been brooding in the community, cf. Schank and Abelson (1975); Woods (1975)

the paradigm of associative networks for representing semantic memory. In his view the 'chunks' of thought should be larger and more structured, and their "factual and procedural content must be more intimately connected". He proposed *frames* as knowledge structures that represent stereotyped *situations*. This meant a move away from the focus on word concepts in semantic nets to more contextual representation of knowledge. Frames are thus presented as part of a theory on the contents of thought (semantic memory), similar to the notion of *script* in Schank and Abelson (1975), used in natural language understanding.

A frame can be regarded as a group of interwoven nodes and relations (somewhat akin to Quillian's planes), but with a fixed structure. The 'top' levels of a frame represent that which is always true for the situation, the lower levels consist of terminals or '*slots*'. These terminals can specify conditions that its assignments (through specific instances or data) must meet. Minsky distinguishes simple conditions – in the form of 'markers' that require straightforward assignments to e.g. (smaller) sub-frames – from complex conditions that specify relations. The terminals of frames are filled with *default* assignments, which may be overridden when a frame is filled by a suitable particular situation.

Frame *systems* are collections of closely related frames. For instance, the effects of possible actions are reflected as transformations between the frames of a system. Frames within a system share the same terminals; this to allow the integration of information from different viewpoints on the same situation. For example, two frames representing the same cube at different angles share some of the faces of the cube.

Where semantic networks could already support a limited form of automatic inference, an important addition of the frame paradigm is the requirement of an *information retrieval network* that supports a standard matching procedure for determining whether a candidate situation fits a frame. This procedure tries to assign values to the frame's markers. In the case of a mismatch, the network should be able to propose a new replacement frame as possible candidate for a match. Matching is not performed solely on the constraints on a frame, but also by the current goals, which are used to determine constraint relevance.

The main contribution of the frame-based view was that it fixed a knowledge representation *perspective*. Semantic nets could be used to represent anything – not just word concepts – and were in many ways equivalent to generic graph representations. The frame proposal fixes the perspective on descriptions of *situations* in general, and *objects* and *processes* in a situation in particular. Minsky (1975) discusses how frames can be used to represent a wide variety of domains – vision, language understanding, memory – without compromising this perspective. His proposal is often viewed in line with the development of *object oriented programming*.

Furthermore, the frame-based approach incorporates a view on the *manipulation* of knowledge, i.e. the transitions between frames in a frame system. This effectively introduced the *reuse* of information as a knowledge organising principle. Lastly, it envisioned a procedure for *matching* specific situations to candidate descriptions in frames, and introduced *defaults* for dealing with incomplete knowledge.

### 2.2.5 Frame Languages

Research in the late seventies produced a number of – what in retrospect could be called – frame based KR languages. Not because they were explicitly developed to define Minsky's original frames (they were not), but because they share its emphasis on interrelated, *internally* structured concepts as primary language primitive.

**Knowledge Representation Language (KRL)** The Knowledge Representation Language (KRL), developed by Bobrow and Winograd (1976), was built on the idea that knowledge is organised around conceptual entities with associated *descriptions* and *procedures*. In their view, a KR language should be independent from the processing strategies or representations of a particular domain. It must provide a flexible set of underlying tools.

KRL descriptions represent partial knowledge about an entity, and can consist of multiple *descriptors* that can be grouped to capture differing viewpoints on the entity. KRL's descriptions are by *comparison* to a known entity (the *prototype*), extended with a further specification of the described entity. The prototype provides a *perspective* from which to view the object being described. The description of a concept entity can combine different *modes* of description (Bobrow and Winograd, 1976, p. 6), such as category membership, stating a relationship, or role in a complex object or event etc.

Reasoning in KRL is done by way of a *process of recognition* where newly introduced objects are compared to stored sets of prototypes. Specialised reasoning strategies can be attached to these prototypes in the form of *procedural attachments*. These procedures could be called depending various triggers (on classes) or traps (on objects) such as goal directed procedure calls (*servant* procedures) and side-effects of system actions (*demon* procedures). Such procedural attachments are coined procedural properties, as opposed to declarative properties.

Bobrow and Winograd make a strong claim that "it is quite possible ... for an object to be represented in a knowledge system only through a set of such comparisons" between prototypes (Bobrow and Winograd, 1976, p.7). The definition of an object is wholly contained within the system, but also functionally complete with respect to that definition as the system can answer any relevant query about it. This represents a fundamental difference in spirit between the KRL notion of representation and standard logical representations. Because the definition of an object is in terms of other objects, and vice versa, and its position within that network of comparisons is determined by a standard inference mechanism, it is the inference mechanism that determines the *meaning* of an object.

**Structured Inheritance Networks (SI-Nets)** Another frame-like language, the Structured Inheritance Networks (SI-Nets) of Brachman (1979) were an attempt to define an epistemologically well-founded class of KR languages (see Section 2.3.2): granted that we distinguish concepts and relations, how can we account for the apparent meaning of concepts that determines their position within a network? The most prominent of these languages, KL-ONE (Brachman, 1979; Brachman and Schmolze, 1985), is organised around *concepts*. Concepts are *intensional*, and can represent objects, attributes and relations in a

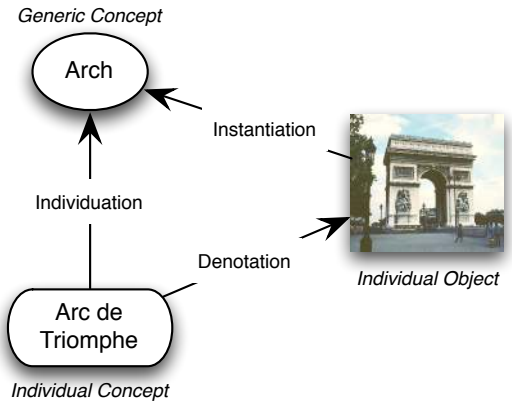


Figure 2.4: Instantiation, individuation and denotation, from Brachman (1979)

domain. Brachman furthermore distinguishes between *generic* and *individual* concepts:[2]

**Generic Concept**
represents a class of individuals by describing a prototypical member of the class.

**Individual Concept**
represents an individual object, relation or attribute by *individuating* more general concepts.

**Individual Object**
(or *instance*) is an object in the actual world that *instantiates* a Generic Concept, and is *denoted* by an Individual Concept, see Figure 2.4.

Similar to KRL entities, KL-ONE concepts are *structured* objects. They are described by *role/filler* descriptions, the 'slots' of the concept. These determine the type of entity that can fill the role, the number of fillers and the importance (modality) of the role. Where KRL uses five *modes* of description, KL-ONE is more abstract and distinguishes three role modality *types* – inherent, derivable and obligatory – that enable a modeller to distinguish between information needed for recognition and more neutral descriptions.

KL-ONE supports procedural attachments, but distinguishes *meta* descriptions – meta statements on concepts using KL-ONE's formalism – from *interpretive* attachments. The former, so-called *structural* descriptions (SD), can be used to prescribe the way in which role fillers interact for any individual; SD's relate two or more roles. The latter are similar to KRL's procedural attachments, and can be expressed using the interpreter language that implements KL-ONE itself (INTERLISP).

In summary, frame-based KR languages introduced a number of knowledge structuring principles that were absent in earlier semantic nets. They

[2]The distinction is similar to that between types (generic concepts), tokens (individual objects) and occurrences (individual concepts) in philosophy.

focus on structured concepts, which are defined by simple or very elaborate structural or procedural descriptions in terms of other concepts. Concepts are either generic or individual, and are clearly distinguished from their real world counterparts. The meaning of a concept is determined by its position within the network of other concepts, which is enforced by a standard inference mechanism based on the descriptions of the concept. Because of the combination of conceptual and procedural (production rule-like) primitives, languages such as KL-ONE and KRL are sometimes called *hybrid* systems.

## 2.3 Epistemology

"One man's ceiling is another man's floor"

*Paul Simon, 1973*

Frame based languages proved to be a significant improvement over other semantic networks.[3] They fixed a paradigm which was more rigourous and better suited for representing knowledge. The development of these languages was not only given in by the cognitive psychology argument of Minsky (1975), but also (and perhaps more importantly) by the growing awareness of the need to have a clear definition of what a knowledge representation *is*.

The interaction between psychological insights and knowledge representation practice led to two fundamental questions:

1. What is the relation between a machine representation and the thing (domain, body of knowledge) it denotes? and,

2. How does the representation language relate to the representation itself?

So, while at first developers of both semantic and procedural knowledge representations were primarily concerned with the *psychological plausibility* of their models, the proliferation of semantic nets and frame based languages sparkled growing concern about the *epistemological status* of knowledge, representation, and the KR languages themselves.

In his 1980 inaugural presidential address to the AAAI[4] Newell (1982) discussed exactly this topic: the nature of knowledge, and in particular the relation between knowledge and representation. In his view, the latter is used precisely and clearly in computer science while the former is often used informally. He identifies a problem with representation, in that it is attributed a 'somewhat magical' role.

> "What is indicative of underlying difficulties is our inclination to treat representation like a *homunculus* as the locus of *real* intelligence."
>
> (Newell, 1982, p.90)

[3] The above quote was taken from Brachman (1979).

[4] American Association of Artificial Intelligence

Table 2.2: Computer System Levels (Newell, 1982)

| Level | Description |
|---|---|
| *Knowledge Level* | Knowledge and rationality |
| *Symbol Level* | Symbols and operations (also *program* level) |
| *Logic Level* | Boolean logic switches (e.g. AND/OR/XOR gates, consists of the *register-transfer sublevel* and *logic circuit sublevel*) |
| *Circuit Level* | Circuits, connections, currents |
| *Device Level* | Physical description |

The most salient task in AI is identifying the proper representation of a problem. It can make the difference between combinatorial and directed problem solving: "... the crux for AI is that no one has been able to formulate in a reasonable way the problem of finding the good representation, so that it can be tackled by an AI system." (Newell, 1982, p.3). The capability to find the proper representation apparently requires some special kind of intelligence.

What epistemological adequacy is, turned out to differ widely, as we have seen in the previous section. McCarthy and Hayes propose to construct a practical philosophical system, based on our common sense understanding of the world. For semantic network-adepts, epistemological adequacy equates to psychological plausibility. But even the criterion of psychological plausibility is not suitably specific to distinguish between production systems and network representations. All three proposals, the logic-based, production-based and network approach, aim to answer the two fundamental questions raised at the start of this section. Although they formulate some description on what a knowledge representation should contain and how it relates to the world it represents, this description remains vague: none of them clearly defines a framework for this relation. And secondly, they do not give an answer to how knowledge relates to the language it is represented in: what are the primitives of knowledge representation?

### 2.3.1 Knowledge and Representation

In his discussion on the nature of knowledge, Newell (1982) presented the *knowledge* level as a *computer system level*. The execution of computer program code is made possible by its translation to physical operations on a circuit board. This translation passes through a number of steps, or 'levels' at which the program can be expressed (e.g. from java code at the symbol level, to java byte code to processor instructions etc.), see Table 2.2. Levels have a *medium*, the *system* it is used to express, primitive processing *components* and guidelines for their *composition*, and a definition of how system *behaviour* depends on the behaviour and structure of components.

Every computer system level can be defined autonomously – without reference to another level – and is reducible to a lower level. Because a description of a system at some level does not imply a description at a higher level, these

levels are not levels of abstraction: A level is rather a specialisation of the class of systems that can be described at the level directly below it. Computer system levels are concrete and really exist, they are "a reflection of the nature of the physical world" (p. 98). Newell postulated the existence of the knowledge level as a hypothesis:

> **The Knowledge Level Hypothesis**. There exists a distinct computer systems level, lying immediately above the symbol level, which is characterised by knowledge as the medium and the principle of rationality as the law of behaviour.
>
> (Newell, 1982, p.99)

The question is, how does the *knowledge* level fit into the rather technical framework of levels in a computer system? The idea is that when we perform knowledge representation – both procedural and declarative – we express our rather abstract, implicit notions of knowledge manipulating systems (people, or rather *agents*) in terms of a symbol level system. This includes the definition of the medium (knowledge), components (actions, goals), and laws of behaviour (rationality) prevalent at this level. Newell (1982) insisted that a knowledge level description of a system is not just a matter of *treating* a system as a rational agent as in the *intentional stance* of Dennett (1987). But rather that the level exists and behaves in the same way as any of the other computer system levels. The term 'knowledge level' is often used to describe representations of knowledge in terms of concepts, tasks, goals etc. The representation is said to be 'at' the knowledge level. For Newell, however, the knowledge level is the knowledge itself, a representation will always be 'at' the symbol level (Newell, 1993).

One implication of this perspective is that a knowledge representation is to be regarded as truly a representation of *knowledge* and not a representation of physical, philosophical or psychological *reality*. Though, by its very nature knowledge is itself some representation of *a* reality. The relations between concepts and individual objects of KL-ONE in Figure 2.4 are in fact reflections of our knowledge of that object.

Figure 2.5 is an extended version of Brachman's triangle and illustrates the relation between a knowledge representation and reality. At the far right are the individual objects that exist in reality, our knowledge of reality is cast both in terms of knowledge of these individuals as individual *concepts* and as generalisations over these individuals. A knowledge *representation* is constructed in a similar fashion. Individual concepts in representation are denoted by our knowledge of individual concepts; generic concept representations are individuated by these individual concepts. However, we can also choose to represent a generalisation over the generic concepts in human knowledge, these *meta concepts* individuate representations of generic concepts, and are instantiated by actual generic knowledge. Finally, a knowledge representation language provides constructs that allow us to formulate the concepts in our representation through instantiation.

Although it is clear that all knowledge representation occurs *by proxy* of our knowledge of reality, it is not always practical to take the separation between knowledge and reality explicitly into account. If a system is to represent reality

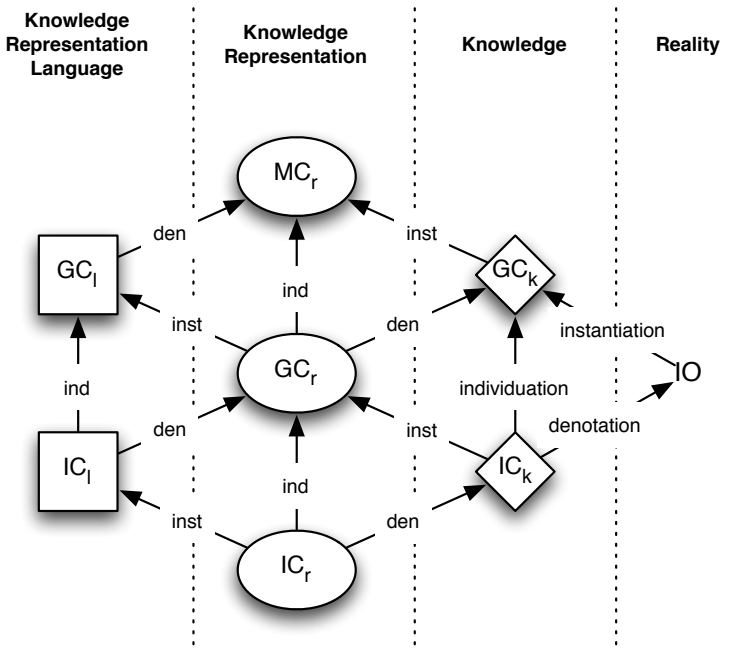


Figure 2.5: Relation between a representation and the world.

as we understand it, there is no more useful and well-tuned proxy than the mental models of reality we use and live by every day.

### 2.3.2 Representation and Language

The introduction of the knowledge level helps us to put the representation of knowledge into perspective, but does not address the suitability issue of the knowledge representation *languages* of Section 2.2.

**Levels** In his review of lessons learned in semantic nets, Brachman (1979) identified five distinct groups of primitive types used in these languages. He considered each of these groups to stand for a particular viewpoint, or conceptual 'level'. Any network, he argued, can be "analysed in terms of any of the levels" (p.27). In other words, a concept expressed in a language at one level, can be understood and expressed at all other levels as well. On the other hand, an interpreter usually commits to support only one of these sets.

At the *implementational* level, semantic nets are mere graphs, data structures where links are pointers and nodes are destinations for links. The *logical* level emerged in reaction to criticism that semantic nets did not have formal semantics. It perceives semantic nets as a convenient depiction of predicates or propositions (the nodes) and the logical relationships between them (the links). Originally, however, semantic nets were meant to capture the meaning of word concepts. At this *conceptual* level, links are case relations between nodes rep-

Table 2.3: Levels of Semantic Networks (Brachman, 1979)

| **Level** | **Primitives** |
|---|---|
| Implementational | Atoms, pointers |
| Logical* | Propositions, predicates, logical operators |
| Epistemological | Concept types, conceptual sub pieces, inheritance and structuring relations |
| Conceptual | Semantic or conceptual relations (cases), primitive objects and actions |
| Linguistic | Arbitrary concepts, words, expressions |

* Note that the logic*al* Level of Brachman is *not* the same as Newell's Logic Level

resenting word senses. Here, the primitives are less neutral, and encompass conceptual elements and relations, such as action types and cases (thematic roles) respectively. Not always are these primitives explicitly defined as part of the semantic net language, but on the whole the relations do have this flavour. One level higher, nodes and links are language dependent. *Linguistic* level networks are composed of arbitrary relations and nodes that exist in a domain. Each consecutive level adds a commitment to a particular interpretation of the structure of the world.

In line with the criticism of Woods (1975), who urged the consideration of the semantics of KR languages, and Minsky (1975), who argued for structured concepts, Brachman postulates that part of the promiscuity of semantic network languages lies in the absence of an intermediate level between the logical and conceptual levels. He proposed the introduction of an *epistemological* level which allows the definition of knowledge-*structuring* primitives as opposed to knowledge primitives:

> "The formal structure of conceptual units and their interrelationships as *conceptual units* (independent of any knowledge expressed therein) forms what could be called an *epistemology*."
>
> (Brachman, 1979, p.30)

To illustrate, even while we can argue about *which* properties exist, we can still agree that *properties* exist. See table 2.3 for an overview of Brachman's levels. Perhaps his levels are best understood as levels of detail or abstraction. When regarding a linguistic level representation, using plain English words etc., we can zoom in to see the case structure and concepts that underlie the language. If we then zoom in again, we can view the internal structure of these concepts and what makes that they can be related in certain ways. Not very surprisingly, his KL-ONE is designed for representing knowledge at the epistemological level.

We must be careful, however, not to misinterpret the analysis Brachman made as a deconstruction of layers in semantic-network based knowledge rep-

resentation only. In fact, it remains rather unclear *what* Brachman believes to be at a level. His description leaves room for the following alternative interpretations:

> **Language**
> A KR language can be designed to be adequate for the representation of knowledge using primitives at a particular level.
>
> **Knowledge**
> The knowledge represented using a KR language can be of a type corresponding to one of the levels. For instance, it is quite common to describe concepts using some form of logic, but we can just as readily represent logical primitives *as concepts*.

In the first sense, the knowledge *primitives* determine the level of a language; where in the second sense the level describes the *kind* of knowledge expressed in a model. The two interpretations of the levels are not wholly unrelated, and Brachman formulates a number of requirements for KR languages to adequately support the representation of knowledge at a particular level. Firstly, a language should be *neutral* with respect to knowledge at the level above it. Secondly, it should be *adequate* for supporting the level above it, i.e. it should be expressive enough to account for knowledge at that higher level. And thirdly, it should have well defined *semantics*: it should prescribe legal operations, and provide a formal definition of its primitives.

**Types** For a long time, the expert systems field seemed to steer clear of the epistemological crisis of declarative knowledge representation. And indeed, the levels presented in the previous section do not particularly fit the heuristic perspective of production systems. Although the PSI architecture includes a 'memory' in which knowledge of the world is stored declaratively, these symbol structures are usually very simple hierarchies with limited semantics, and were used chiefly as database-like place holders for instance data.

All of this changed when it became clear that production rule systems were not particularly usable in settings other than which they were designed for. This realisation emerged when Clancey (1983) attempted to use MYCIN rules in the GUIDON tutoring program, i.e. to "transfer back" expert knowledge from a rule base. The idea was to use the rules to explain each step in the diagnostic reasoning process. As the original design goal of MYCIN was for it to be built using a simple mechanism for representing heuristics that would support explanations and advice, it seemed at first that this educational use would be relatively straightforward. It turned out not to be. Merely using MYCIN's built in explanation facility did not work as expected. GUIDON was incapable of explaining many rules because of insufficient information about the way the rule base was structured. In order to support a tutoring setting, it is necessary to extract this "compiled knowledge"(Clancey, 1983).

Rules in MYCIN, but in other systems as well, implicitly encode the design rationale behind the way rules are fitted together. Clancey clarifies the different ways in which *design knowledge* is lost when building rules by distinguishing between *goals*, *hypotheses* and *rules*. These three categories are organised in a network of dependencies (see Figure 2.6). Goals are formulated as questions

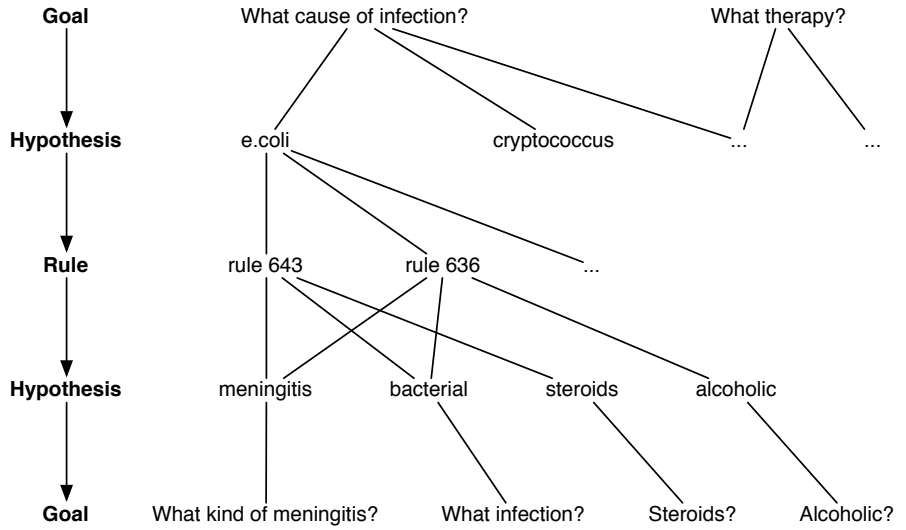


Figure 2.6: Network of goals, hypotheses and rules, adapted from Clancey (1983)

that need to be answered for the system to solve a problem (e.g. making a diagnostic decision). Hypotheses are potential answers to the questions, and are ascertained by rules that have the hypothesis as their consequent. Rules fire, i.e. make their consequent hold, when their antecedent is satisfied. This antecedent is composed of other hypotheses that answer some goal.

The decomposition gives insight in the way the different categories interact when MYCIN traverses the search space. For instance, when it tries to satisfy e.g. the "meningitis" hypothesis, MYCIN will in fact consider all related hypotheses that answer the more general goal "what infection?". The links between these categories are the 'points of flexibility' in a rule representation.

A problem solving *strategy* can be conveyed by making explicit the rationale behind the order of premises in a rule as this affects the order in which goals and hypotheses are pursued. A decision to determine the ordering of hypotheses in a particular way is a *strategic decision*, which can be stated in relatively domain-independent terms. The example given by Clancey is "consider differential broadening factors". Also some level of *structural* knowledge is necessary to "make contact with knowledge of the domain", it provides a "handle" by which a strategy can be applied. For instance, it seems reasonable to investigate common causes of a disease before considering unusual ones.

The *justification* for a rule is captured by the rationale for the connection between the conclusion in the consequent and hypotheses in the antecedent. Justification depends on knowledge of the domain. Clancey (1983) identifies four different types of justification, and consequently four different types of rules (the *domain theory*):

- *Identification* rules use properties of an object to identify it, e.g. "if it walks like a duck and talks like a duck, it is a duck".

- *Causal* rules convey a causal relation between two objects, e.g. "problem causes disease". In fact MYCIN distinguishes several different types of causal relation.
- *World fact* rules capture common sense knowledge about the world, e.g. "if the patient is male, he is not pregnant"
- *Domain fact* rules capture domain specific knowledge on the basis of domain definitions, e.g. "if a drug was administered orally and it is poorly absorbed in the GI tract, then the drug was not administered adequately"

This distinction between strategy and justification proved to be a very potent methodological tool. The structural domain theory can be used to make the strategy explicit that 'indexes' the domain model (the knowledge base). In other words, instead of a hodgepodge of entangled rules, the different components of a production rule system can now be considered separately and relatively independently. All this by a shift from the dominant heuristic perspective to a more epistemological analysis of the types of knowledge involved in production systems.

### 2.3.3 Adequacy

The distinction between epistemological and heuristic adequacy of McCarthy and Hayes (1969) turned out to have a deeper impact than originally envisioned. Although it was primarily presented as the two main goals of AI, it soon turned into a rift in artificial intelligence research between epistemological–declarative (semantic networks) and heuristic–procedural (production system) systems. Because they were treated as very distinct notions, their pursuit has produced very different representation perspectives. But as we have seen, these are really two sides of the same coin.

Firstly, heuristic and epistemic approaches tend to deal with the same *kinds* of knowledge. Albeit in quite divergent ways. Frame descriptions are not purely 'epistemological'; surely the assignment of default slot values based on a 'match' is a form of inference and presupposes some procedure for feeding new information to a knowledge base. Both KRL and KL-ONE are in fact *hybrid* systems and combine declarative concept definitions with procedural attachments. Vice versa, production systems encode identification knowledge and knowledge of facts, the primary issue in declarative approaches, and are thus just as susceptible to the epistemological crisis.

Secondly, the two approaches interact; how can a representation mean anything without some form of standard inference? McCarthy and Hayes's representation is "in such a form that the solution of problems follows from the facts expressed in the representation" (p.5), but how to check whether a solution indeed follows from the representation? In other words, the inference mechanism in frame languages is just a particular heuristic strategy of the kind production systems are intended to capture.

Newell's analysis showed that *both* aspects should be present for *any* knowledge level representation as roles of behaviour on the one hand, and the composition of components on the other. Both Brachman and Clancey hold that KR languages should provide the basic elements that shape our knowledge.

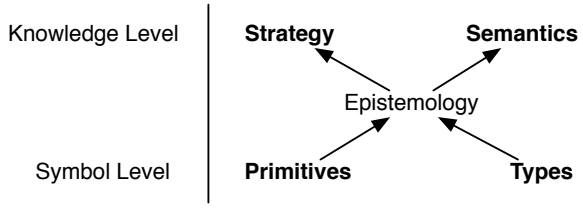


Figure 2.7: Levels of Knowledge Representation

Brachman emphasises the explicit definition of knowledge-structuring primitives. Clancey distinguishes *components* of a knowledge base, the interaction between different types of knowledge.

Furthermore, the separation of heuristics and epistemology is still important, but has been detached from the KR formalisms and has become rather an interaction between strategic and domain knowledge. A strategy 'indexes' domain facts, and the combination of the two constitutes a domain model.

Both Clancey and Brachman thus operate at the epistemological level, and argue that only knowledge representation at this level can account for knowledge level system behaviour. For Clancey the ultimate test is the *reuse* of a knowledge system for a different task (explanation vs. diagnosis: same domain, different strategy), for Brachman this was the built-in appropriateness of a language to represent a particular kind of (higher-level) knowledge.

More than a decade later, Davis et al. (1993) tried to settle the issue once and for all. A lot of the discussion in the field of knowledge engineering is, in their view, caused by a conflation of the different roles played by a knowledge representation. As we have seen, knowledge representation is first, and foremost, a *surrogate* for 'the thing itself' – a particular domain – used to enable reasoning as a simulation of the domain. As a representation cannot cover *all* of the domain, it needs to make a selection of relevant features of the domain and thus fixes a perspective on it. Every representation is therefore a set of *ontological commitments*, a commitment to the terms in which one should think about the domain (the domain and world facts of production systems). This commitment pertains not only to the contents of a particular model, but also to the KR language used. The ontological commitment accumulates in layers. For instance, a language for describing knowledge at the conceptual level commits to the existence of concepts, whereas a logical level language makes no such claims.

As we have seen in the preceding, a well designed representation language includes a standard inference mechanism. This makes a language a *fragmentary theory of intelligent reasoning*; it sanctions heuristic adequacy, and prescribes the way in which an AI system reasons on the basis of some adequately represented domain. Also, in its 'magical' role (Newell (1982), cf. Section 2.3.1) a KR language includes a commitment to a particular *way* of formulating problems that turns it into a *medium for pragmatically efficient computation*. The combination of language and representation is by itself a language which allows us to describe the world in a particular way; it is a *medium of human expression* (Stefik, 1986).

## 2.4 Components of a Knowledge Based System

As described in Section 2.3 the reusability of expert systems can be enhanced by separating the different types of knowledge in a model. In particular, this insight proved valuable in dealing with the problem first encountered by Wilkins and Leibniz in the 17th century: accumulating and representing all knowledge necessary for performing a particular task can be an arduous undertaking. Of the three schools discussed, the expert systems field was the one exposed to such large quantities knowledge, that this step of *knowledge acquisition* (KA) became regarded as worthy of study in its own right.

In early expert system development, models were built in an ad hoc, unprincipled and incremental manner. This lead to models such as that of MYCIN, in which expert knowledge was 'compiled' away, which made them very hard to understand for people other than the original designers. The models could not be reconstructed for other purposes than the one originally intended. Another problem was that this left no other means to check the correctness of a model, than by evaluating system behaviour as a whole. The knowledge acquisition field soon set out to develop a priori ways for ensuring expert system quality. Knowledge representation had become an engineering task, knowledge should be *modelled* rather than merely extracted from experts, and this process of knowledge engineering should be guided by a principled methodology. The methodology guarantees a level of quality by making design decisions explicit (see also Chapter 5).

### 2.4.1 Modelling Principles

In most approaches of the 1990's, knowledge engineering is regarded as a creative activity in which the construction of a knowledge base should be preceded by a modelling step. As in software engineering, the actual implementation of a KBS is guided by a *functional specification*. These approaches had a very strong methodological perspective, and covered every step in the construction of a KBS. From the identification of the purpose of a system and methods for knowledge elicitation and acquisition, to the specification and actual implementation of the system.

For instance, in the KADS methodology (Wielinga et al., 1992; van Heijst et al., 1997) knowledge acquisition is the construction of a knowledge level model (the *knowledge* or *conceptual* model).[5] Knowledge representation is then the implementation of this knowledge-level model in a knowledge base. This *design* model is a symbol level representation and takes into account additional considerations regarding e.g. computational efficiency of the system (See Figure 2.8). These considerations are recorded as *design decisions*. The PROTÉGÉ system of Puerta et al. (1992) adopted a similar approach where an initial knowledge level model was automatically translated into a CLIPS rule base.[6]

[5]Rather than a model consisting of 'concepts', the KADS conceptual model itself has concept status, it is a preliminary, abstract *version* (in another language) of the design model (a system component).

[6]CLIPS: C Language Integrated Production System, see `http://clipsrules.sourceforge.net`

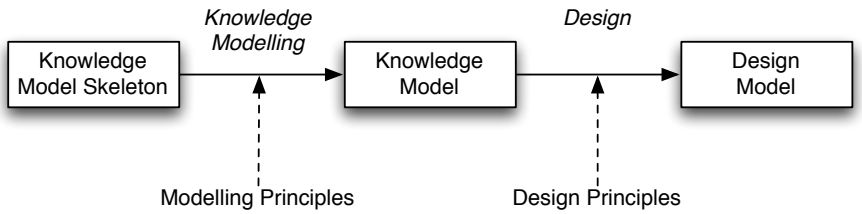


Figure 2.8: Steps in the knowledge engineering process (van Heijst et al., 1997)

Knowledge systems can be described at the knowledge level of Newell (1982), as long as the description is properly abstracted from the structural organisation of knowledge. In other words, the knowledge base (KB) of such a system should be described in terms of its *competencies*: "knowledge is to be characterised entirely *functionally*, in terms of what it does, not *structurally* in terms of physical objects with particular properties and relations" (p. 105). Levesque (1984) describes a functional perspective on KB characterisation. A KB can be treated as an *abstract data type*, where its behaviour is described in terms of a limited set of TELL and ASK methods. The capabilities of a KB are a function of the range of questions it can answer and assertions it can accept. How a KB implements this functionality should be hidden from the rest of the system. The advantage of this functional approach is that the knowledge engineer does not have to take into account considerations of implementation while constructing the knowledge model – at least to a large extent. As a result, the knowledge level model is less biased towards the implementation in a particular knowledge representation language.

Knowledge level models are more accessible to domain experts as they can be constructed and interpreted without in-depth understanding of technicalities. A knowledge level model can therefore help to reduce the *knowledge acquisition bottleneck* (Feigenbaum, 1980): the general difficulty to correctly extract relevant knowledge from an expert into a knowledge base.

Reuse of symbol level representations in the way discussed by Clancey (1983) turned out to be problematic because of the *interaction problem* (Bylander and Chandrasekaran, 1987). The problem that different types of knowledge in a knowledge base cannot be cleanly separated, because the purpose of the KBS, – translated into strategies – influences the way in which the domain theory is structured (recall Newell (1982) in Section 2.3). This is similar to the context dependency of the meaning of concepts. However, as knowledge models are formulated at a more abstract level – relatively untainted by issues of implementation and structure – it was thought that reuse of such models would be feasible. This "limited interaction hypothesis" (Wielinga et al., 1992) assumed that if domain knowledge was represented in a 'well structured and principled manner', general descriptions of methods should be possible. Unfortunately the structure of domain knowledge is often characteristic for a domain, and is not merely a matter of design (Valente et al., 1998).

The development of libraries of *skeletal* models, partially filled-in models of typical use, was thought to be the answer to the *hugeness* problem. Building a KBS requires a significant effort, not only because of the KA bottleneck, but

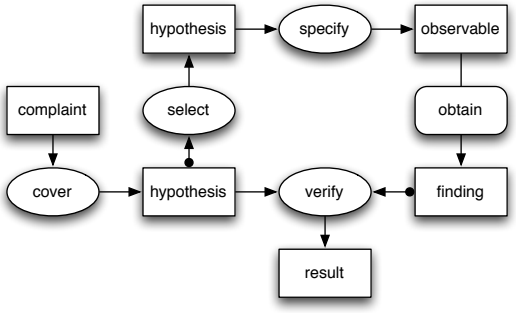


Figure 2.9: The standard *diagnosis* PSM, from Schreiber et al. (2000).

also because expertise domains in general are very large and complex. Skeletal models do not only help to reduce the amount of specification needed, but provide guidance with respect to the knowledge needed to build a KBS as well: they ensure coverage.

To enable knowledge level reuse, the roles played by elements in a model should be identified (Clancey, 1983; Bylander and Chandrasekaran, 1987). Approaches based on *role limiting* facilitate reuse by indexing symbol level representations – executable models – to types of knowledge at the knowledge level (Valente et al., 1998). These models are detailed blueprints for implementation in a system. Other approaches, such as KADS, rather took skeletal models to be sketches of models and provided no direct connection to implementation. The models in KADS supported reuse of 'understanding'.

Role limiting and knowledge typing are accomplished firstly by the separation between heuristics and epistemology.[7] For instance, KADS categorises elements as a type of *control* knowledge or as part of a *domain* theory (See also Figure 2.7). Control knowledge is the knowledge regarding how a system obtains it goals and solves problems. The *Common*KADS approach distinguished two types of control knowledge in expertise models (van Heijst et al., 1997; Valente et al., 1998; Schreiber et al., 2000):

**Task Knowledge**
is a abstract high level description of the decomposition of the goals within a KBS that must be achieved by problem solving.

**Inference Knowledge**
An inference is an primitive problem solving step which is solely defined in terms of its input/output signature. Its internal workings cannot be meaningfully expressed at the knowledge level even though they can perform very complex operations. Inferences can be combined in inference *structures*.

Tasks and inferences are combined in problem solving methods (PSM); these are descriptions of a particular way to perform some task: a PSM expresses a strategy, and reflects the "competence to decompose a task" (Valente et al.,

[7]Note that at the knowledge level, this does not correspond to a distinction between procedural vs. declarative: everything specified at the knowledge level is declarative.

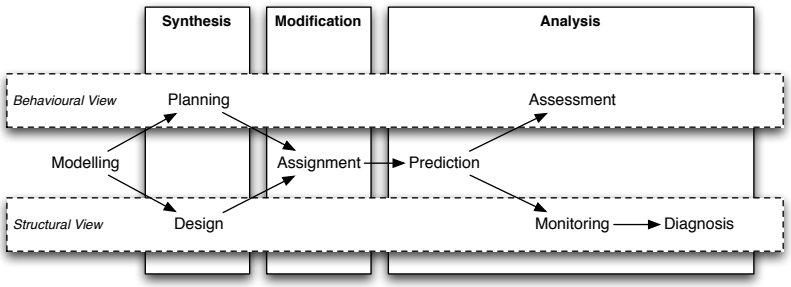


Figure 2.10: A suite of problem types, dependencies and views from Breuker (1994).

1998, p.400). See Figure 2.9 for an example problem solving method. Where role limiting approaches generally used a fixed task hierarchy, PSMs are orthogonal to such a hierarchy and can combine several tasks to perform a more complex one. In other words, if a task hierarchy divides a task into sub tasks, a PSM describes a particular path along those tasks in terms of an inference structure that implements them. Breuker and Van De Velde (1994); Breuker (1997) argue that a library of PSMs should be indexed by a suite of problem types, rather than by a taxonomy of tasks (Breuker, 1994).

Breuker's suite defines an intermediate level between task decompositions and problem solving methods. Central to this view are two standard sequences of problem types part of any task (Figure 2.10). The two sequences can be distinguished by a *behavioural view* and a *structural view*. The former incorporates the problems that deal with the interaction of a system with its environment: planning and assessment. The latter includes problems of design, monitoring and diagnosis and has a more internal perspective.

The interaction problem tells us that control knowledge and domain theory cannot be fully separated in the way assumed in the limited interaction thesis. On the other hand, a too close connection between the two kinds of knowledge severely limits reusability. A library of reusable knowledge components should therefore contain descriptions of domain knowledge: a PSM cannot be specified without at least some reference to domain knowledge. Firstly, a PSM expresses domain knowledge in terms of its role in a problem solving process. This captures the use of domain knowledge as an *epistemology* (see Section 5.5.2). Secondly, this epistemology is connected to its possible role fillers: the categories in the domain that are able to fulfil a role in problem solving. Reuse of PSMs is achieved by connecting categories in the epistemology to their counterparts in a generic domain theory. In short, the domain theory has two roles:

- To index problem solving methods for *reuse*, and in this way
- Communicates the *kinds* of things an expert system 'knows' about.

By the way it is constructed, an expert system cannot do otherwise than simply disregard anything it does not know about. In a way, the incorporation of a particular domain theory is therefore a significant *ontological commitment*,

in the sense of Davis et al. (1993). The domain theory expresses *that which persistently exists* for an expert system: it is an '*ontology*' (see Chapter 4).

## 2.5 Knowledge Representation

The views of KADS, *Common*KADS and the role limiting approaches did not just have impact on knowledge level specification (in the engineering sense) but also influenced the languages used for knowledge representation. While the psychological view and rationale of problem solving was abstracted to a meta-level of problem solving and reasoning strategies, knowledge based systems still require implementation at the symbol level. Even when this implementation is not intended to be *directly reusable*, as in role limiting, it needs to explicitly commit to a level of quality with respect to efficiency and computational properties.

In line with the functional paradigm of Levesque (1984), a knowledge base is in fact a *service* that can function as a black-box component of knowledge-based systems. As Levesque pointed out, to be reusable, such a component must be well described: it should guarantee answers to a specific set of tell/ask queries. This way, the different knowledge types of knowledge level specification can be instantiated as separate components of a knowledge based system *architecture*. This means that although *reasoning*, and problem solving in particular, is often a non-monotonic affair (McCarthy, 1980) where hypotheses are frequently asserted and retracted, *inference* does not have to be.[8]

Because KL-ONE like languages were already the most targeted to a specific type of knowledge, they gradually took on the singular role of expressing the domain theory of knowledge based systems.[9] This required the fine-tuning of standard inference over the *structural descriptions* that determine the position of a concept definition in a network: *classification*. At the same time, the logic approach first advocated by McCarthy and Hayes (1969) found its way back into mainstream knowledge engineering. Classification was specified as a form of *logical inference*. The functional specification advocated by Levesque was thereby underpinned by a logical formalism that specified the exact semantics of the representation language used by the knowledge component (Levesque and Brachman, 1985).

Levesque and Brachman (1987) pointed at a trade off between the expressive power and computational efficiency of representation formalisms. Inference on a highly expressive language will generally be inefficient, or even undecidable, while limitations on expressive power can ensure tractability. Relaxing either of these requirements would result in a system whose answers are not dependable. Its conclusions may not follow logically from the theory expressed by the knowledge base (soundness), or it does not give all possible answers (completeness). Levesque and Brachman (1987) therefore propose what

[8] For sure, in some applications a non-monotonic knowledge base can be beneficial for performance reasons as not the entire knowledge base needs to be re-evaluated whenever information is added or removed. However, non-monotonicity in general is no requirement as e.g. recent progress in decidable incremental reasoning over monotonic description logics shows (Parsia et al., 2006).

[9] Although logic programming languages such as Prolog were very popular for the representation of domain theories, their order dependence makes that a Prolog representation will always be slightly tainted by control knowledge.

Doyle and Patil (1991) call the *restricted language thesis*:

> "...general purpose knowledge representation systems should restrict their languages by omitting constructs which require non-polynomial (or otherwise unacceptably long) worst-case response times for correct classification of concepts"
>
> (Doyle and Patil, 1991, p.3)

However, as Doyle and Patil (1991) argue, this requirement cannot be posed in general for all representation languages used in knowledge based systems. In many applications, undecidable reasoning has no serious consequences and languages should therefore be evaluated on the basis of whether they "provide the rational or optimal conclusions rather than the logically sound conclusions" (Doyle and Patil, 1991, p.6). Although this is a fair point for knowledge based systems *as a whole*, it reintroduces the epistemological promiscuity of rationality at the implementation level, and thereby the interaction problem of Bylander and Chandrasekaran.

Successors of KL-ONE such as NIKL (Moser, 1983), KL-TWO (Vilain, 1984) and LOOM (MacGregor and Bates, 1987) and the Frame Language (Levesque and Brachman, 1987, FL), KANDOR (Patel-Schneider, 1984), KRYPTON (Brachman, 1983) and CLASSIC (Borgida et al., 1989; Brachman et al., 1991) were often still *hybrid* systems that combined terminological reasoning with procedural attachments. Once class membership or subsumption is established using *triggers* (Brachman et al., 1991), additional rules could fire that assigned default values to certain properties. Furthermore, Baader and Hollunder (1991) pointed out that these systems often implemented sound, but *incomplete* subsumption algorithms. The main reason was that sound *and* complete algorithms were only known for small and relatively inexpressive languages. Additionally, for many languages the subsumption problem was at least NP-hard, which meant that complete implementations would be intractable, while incomplete algorithms could be polynomial. Tractability is important for practical applications with often critical response time requirements.

### 2.5.1 Description Logics

The quest for more expressive but *decidable* combinations of language features became ever more prominent in the development of *description logics* (DL). These logics are often regarded as the direct successor of frame-based languages. However, where in frame-based systems descriptions (or descriptors) are secondary to the concepts they describe, in DL the descriptions themselves are first-class citizens. Current DLs have thus shifted away from concept-centered representation to description or *axiom* centred semantics. Concepts merely group together multiple descriptions to create some cognitively plausible aggregate. This paradigm shift is understandable, as frame-based systems, and early DL-systems, were not entirely 'clean' regarding the separation between concept and description. The $\mathcal{KRIS}$[10] system of Baader and Hollunder (1991) is one of the first sound and complete implementations of an expressive but decidable description logic.

[10] Knowledge Representation and Inference System

Description logics are fragments of first order logic (FOL) usually defined by means of a *model theoretic* semantics, i.e. its semantics is based on the interpretation of a language by means of set-theoretic structures. A sentence in such a language can only be made true or false given some *interpretation* of that sentence in terms of actual entities. For instance, the sentence "The person sits on the chair" can only be determined to be true given a mapping to a corresponding situation in reality, the *domain*, e.g. given a mapping from 'person' to an actual person, from 'chair' to an actual chair, and from 'sits' to an actual sitting relation between *that* person and *that* chair. An interpretation that makes the sentence true is said to be a *model* of that sentence. In terms of Brachman (1979) (cf. Figure 2.4), the interpretation of some sentence is the denotation relation between generic and individual concepts on the one hand, and individual objects and sets of objects in the domain on the other. See Figure 2.11 for a schematic depiction of the relation between concepts, individuals and the domain.

More formally, in DL a model consists of a domain $\Delta^{\mathcal{I}}$ and an interpretation function $\cdot^{\mathcal{I}}$ that maps individual names to elements in the domain (individual objects), class names to subsets of the domain, and property names to binary relations on the domain. The vocabulary $N$ of a DL knowledge base $\mathcal{K}$ correspondingly is a tuple $\{N_I, N_C, N_R\}$ consisting of a set of individual names $N_I$, a set of class names $N_C$, and a set of role names $N_R$. To give an example, applying the interpretation function to an individual $o \in N_I$ must result in a member of the domain: $o^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. Similarly, the interpretation of a class $\mathsf{C} \in N_C$ is a subset of the domain ($\mathsf{C}^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$), and the interpretation of a property $\mathsf{R} \in N_R$ is a subset of the set of all possible object pairs in the domain: $\mathsf{R}^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The meaning of axioms in a DL knowledge base is given by the relations between individuals, classes and properties in that knowledge base and corresponding constraints on models. For instance, if $\mathsf{A} \sqsubseteq \mathsf{B}$ ($\mathsf{A}$ is a subclass of $\mathsf{B}$), then $\mathsf{A}^{\mathcal{I}} \subseteq \mathsf{B}^{\mathcal{I}}$ for all models of the knowledge base. In fact, this meaning does not depend on any meaning of objects in the domain, or on which objects make up the domain. Rather, the meaning of an axiom arises in its relation to the other axioms in a knowledge base: every DL knowledge base consists at least of the two classes top ($\top$) and bottom ($\bot$) which are defined as $\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$ and $\bot^{\mathcal{I}} = \emptyset$, respectively. The meaning of a DL knowledge base "derives from features and relations that are common to all possible models" (Horrocks et al., 2003).

The way in which the semantics of DL axioms is defined has several important implications. First, and foremost, every assertion in the knowledge base should be seen as a restriction on the possible models of that knowledge base. This means, conversely, that even though something might not have been explicitly stated, it may still be assumed to hold (i.e. be a model of the knowledge base). Description logics thus adopt the *open world assumption* which puts a rather counter-intuitive spin on negation: an axiom is only 'false' iff it has no model, i.e. that model is not a model of the knowledge base. Negation is therefore quite hard to enforce in DL.

A second implication is that there exists no direct link between a knowledge base and a domain. Although a knowledge base is defined in terms of the interpretation to the domain, determining adequacy of the knowledge base can only be determined internally, i.e. by 1) inferring that some axiom *can not* have a model (there cannot exist a (set of) objects that can make the axiom true)

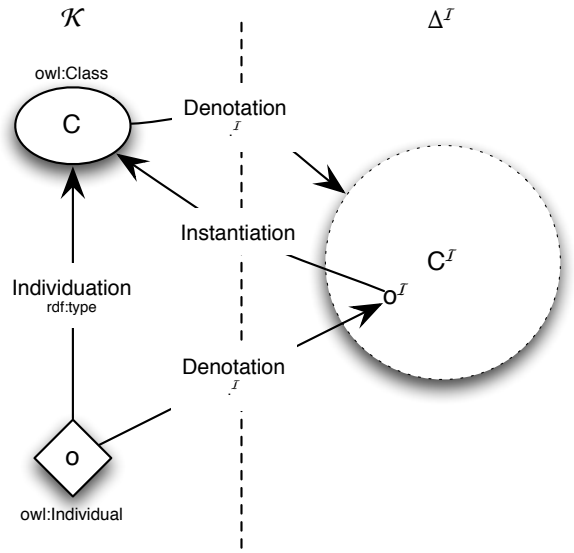


Figure 2.11: Instantiation, individuation and denotation in DL

or by 2) explicit assertions about individuals that relate directly to individual objects in the domain. Whether the latter relation holds or not is up to an observer external to the knowledge base. Related to this, is the fact that DLs do not necessarily adopt the *unique name assumption*, the assumption that an individual object is denoted by exactly one individual in the knowledge base: any number of individuals may share the same model.

As discussed earlier, the balance between computational complexity and decidability on the one hand, and expressiveness on the other plays a prominent role in description logics research (Baader et al., 2003). As a result, various description logics can be constructed out of several basic building blocks, for which the effect on combinations is well known.[11] These building blocks each have an associated letter, the combination of which gives the name of a particular description logic (see Table 2.4). For example, the DL $\mathcal{SHIQ}(d)$ is the attributive language with complex concept negation, transitive properties, inverse properties, qualified cardinality restrictions and datatype properties.

The emphasis on computational efficiency has furthermore lead to a distinction between axioms in the terminological, role and assertional boxes:

**Terminological Box (TBox)**
Contains class axioms, the generic concepts of Brachman (1979).

**Role Box (RBox)**
Contains descriptions of the characteristics of properties and the relations between them.

[11] See e.g. the Description Logics Complexity Navigator at `http://www.cs.man.ac.uk/~ezolin/dl/`.

**Assertional Box (ABox)**
Contains the assertions about individuals, the individual concepts of Brachman (1979).

Although the distinction between these boxes is not formally relevant, separating the various types of reasoning – classification for the TBox and realisation for the ABox – significantly improves reasoner performance (Levesque and Brachman, 1987). For instance, the use of class axioms such as *nominals*, that define a class by enumerating its individual members and thus cross the boundary between TBox and ABox, can have a remarkable effect on reasoning time (Klarman et al., 2008).

## 2.6 Discussion

As the issues raised in this chapter show, knowledge representation is not a trivial task and from its rise in the late sixties, the field underwent numerous changes. We discussed the computational crisis of uncontrolled logical inference in theorem proving, the debates on heuristic and epistemological adequacy, the status and psychological plausibility of knowledge representation languages, the epistemological promiscuity of expert system implementations and the rise of an *engineering* perspective on knowledge based systems development.

Although the three schools of knowledge representation (Section 2.2) embarked on distinct journeys, they each have a distinct role in this knowledge engineering view. The procedural perspective of production systems allows to express rational control in knowledge based reasoning, the semantic network paradigm grew into robust languages for expressing domain theories, and logic returned in full swing as means to define the semantics and guarantee correct inference of implicit knowledge expressed using these languages.

The most important lesson learned is certainly that though human experts may be very successful in applying highly heterogeneous knowledge, directly mimicking this human expertise has a serious detrimental effect on knowledge based systems development, maintenance and performance. Knowledge based systems should be *specified* at the knowledge level. This specification contains a description of the different types of knowledge involved in the system, and how they interact. The most prominent distinction can be made between the *domain theory* of a system – what it knows *about* – and the *control knowledge* that expresses the rationale of reasoning over the domain. Control knowledge consists of a decomposition of tasks and inference types, the performance of which can be specified by generic *problem solving methods*. Inference types are characterised by an input/output signature and can be implemented by separate knowledge *components* that have a matching *functional specification*. The different types of knowledge in an implemented knowledge based system operate at different levels: control knowledge is applied at a meta level with respect to the inferences performed by knowledge components.

The functional specification of knowledge system components is most vigourously adopted by *description logics* systems, the successors of KL-ONE like frame-based languages. Description logics are a family of decidable formal languages designed to express concepts, relations and their instances and are

thus naturally suited for expressing the domain theory of knowledge based systems. DL algorithms are optimised for calculating the inferred subsumption hierarchy, class satisfiability and class membership of instances (realisation).

Description logics research continued throughout the nineties, but played a relatively modest role until it entered a global stage when DL became the formalism of choice for knowledge representation on the *Semantic Web* (Berners-Lee, 1999). Arguably, the development of a web-based knowledge representation language is not trivial. Chapter 3 discusses the trade-offs and restrictions imposed on languages for the semantic web, and gives an overview of the resulting Web Ontology Language (Bechhofer et al., 2004, OWL).

Davis et al. (1993) characterised the domain theory of knowledge based systems as its *ontology*, a term that increased in popularity throughout the nineties. Although the exact meaning of this term is not directly relevant for the discussion of OWL as representation language in Chapter 3, ontologies do have a specific role to play in knowledge engineering. Chapter 4 discusses the various, often confusing, interpretations of what ontologies are, and gives a characterisation of an ontology as knowledge representation artefact.

Table 2.4: DL abbreviations

| Abbreviation | Description |
|---|---|
| $\mathcal{AL}$ | Attributive Language:<br>• Atomic negation (negation of concepts that do not appear on the left hand side of axioms)<br>• Concept intersection<br>• Universal restrictions<br>• Limited existential quantification (restrictions that only have fillers of owl:Thing) |
| $\mathcal{FL}^-$ | A sub-language of $\mathcal{AL}$, without atomic negation |
| $\mathcal{FL}_o$ | A sub-language of $\mathcal{FL}^-$, without limited existential quantification |
| $\mathcal{C}$ | Complex concept negation |
| $\mathcal{S}$ | $\mathcal{AL}$ and $\mathcal{C}$ with transitive properties. In $\mathcal{SROIQ}$: some additional features related to the RBox |
| $\mathcal{H}$ | Role hierarchy (subproperties) |
| $\mathcal{R}$ | Limited complex role inclusion axioms; reflexivity and irreflexivity; role disjointness. |
| $\mathcal{O}$ | Nominals, i.e. enumerated classes and object value restrictions |
| $\mathcal{I}$ | Inverse properties |
| $\mathcal{N}$ | Cardinality restrictions |
| $\mathcal{Q}$ | Qualified cardinality restrctions |
| $\mathcal{F}$ | Functional properties |
| $\mathcal{E}$ | Full existential quantification |
| $\mathcal{U}$ | Concept union |
| $(D)$ | Datatype properties, data values and datatypes |

Source: http://en.wikipedia.org/wiki/Description_logic

# Chapter 3

# Semantic Web

> "He could spell his own name WOL, and he could spell Tuesday so that you knew it wasn't Wednesday but his spelling goes all to pieces over delicate words like measles and buttered toast."
>
> *About Owl, in 'Winnie the Pooh' by A.A.Milne*

## 3.1 Introduction

In November 2001, the WebOnt working group set out to develop a new Web Ontology Language (OWL, Bechhofer et al. (2004)) to be used for knowledge representation in a form that it could be shared across the web.[1] The working group was tasked with the development of:[2]

> "A Web ontology language, that builds on current Web languages that allow the specification of classes and subclasses, properties and subproperties (such as RDFS), but which extends these constructs to allow more complex relationships between entities including: means to limit the properties of classes with respect to number and type, means to infer that items with various properties are members of a particular class, a well-defined model of property inheritance, and similar semantic extensions to the base languages."
>
> WebOnt WG Charter

The Web Ontology Language is tightly interwoven with the development of the *semantic web*, first described in Berners-Lee (1999); Berners-Lee et al. (2001). The purpose of the Semantic Web is to advance the machine interpretability

[1] The working group disliked the proper acronym "WOL" and decided to call the language "OWL". This decision was backed by the extraordinary abilities of the wise friend of Winnie the Pooh: Owl.

[2] `http://www.w3.org/2002/11/swv2/charters/WebOntologyCharter`

of information stored on the web. Not just for improving the accessibility and communication of this information to humans, but primarily for *knowledge* exchange between *automated web services*. More importantly, semantic web technology should enable open and unattended sharing of this knowledge. A web-based knowledge representation language should take into account the distributed nature of the web, and is necessarily subject to other constraints than a language that is not exposed in that way: how does this trade off work out in practice?

In this chapter I give an overview of the technical characteristics of semantic web languages, and the trade-offs they imply for web-based knowledge representation. Central to this discussion is the Web Ontology Language, which is in many ways a natural step up the evolutionary ladder for the terminological knowledge representation languages of the 80's and 90's. Although it can be said that a significant body of documentation on OWL already exists, this has either the nature of reference guide or overview (Bechhofer et al., 2004; McGuinness and van Harmelen, 2004), a technical specification (Patel-Schneider et al., 2004) or rather entry-level guides and walkthroughs (Antoniou and van Harmelen, 2004; Smith et al., 2004; Horridge et al., 2007). This chapter aims to lay bare the rationale and limitations of the language features of OWL, both practical (Antoniou and van Harmelen, 2003) and theoretical (Horrocks and Patel-Schneider, 2003). This view plays a central role in the description of *design patterns* in Chapter 7.

The fact that OWL is called a web *ontology* language, obfuscates rather than emphasises its main character as a highly expressive description logic. For all practical purposes, this chapter will simply use the term 'ontology' to denote a terminological knowledge representation (or domain theory) expressed using the web ontology language. Chapter 4 discusses the ambiguity of the term 'ontology', and sheds light on the historical reasons for giving this language the name "OWL".

### 3.1.1 The Web

The web as it exists from roughly 1993 on is not much more than a network of inter linked pages. For sure, its success is very much based on this lightweight approach: it uses a simple, networked architecture which is easily extensible and immune to incorrect or incomplete data, links and mark-up. Information is presented in a human understandable way: by means of texts, pictures, layout etc. Legacy web languages such as HTML[3] and CSS[4] are specifically targeted to facilitate the disclosure of information in a way suitable for human consumption. In a way, more recent developments such as Ajax[5] take this approach even further by making web pages more responsive to human interaction.

The primary means of navigating the web is by following hyperlinks between pages, form filling and the use of search engines such as Google and Yahoo Search. These search engines use a combination of natural language processing (NLP), network analysis and hit-counts to determine the rank of a page in search results. Nonetheless, still even the best engines suffer from

[3]Hypertext Markup Language, `http://www.w3.org/html/`
[4]Cascading Style Sheets, `http://www.w3.org/Style/CSS/`
[5]Asynchronous Javascript and XML

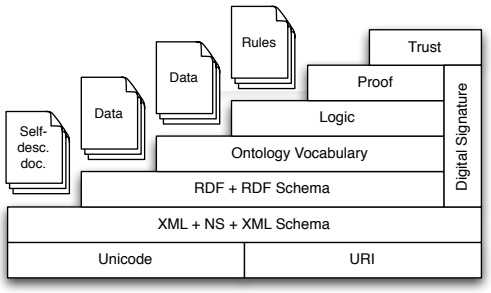


Figure 3.1: The SemanticWeb layer cake

either a high recall (low precision) or low or no recall at all (Antoniou and van Harmelen, 2004). Both search queries and results are language dependent, i.e. a search using one language will not reveal relevant results in another. Furthermore search results are single webpages, whereas information relevant to the query might be distributed across many. It is therefore unsurprising that companies spend huge sums to improve the visibility of their pages in these search engines. According to a Forrester report, spending will increase from today's 4.5 billion to 8.1 billion euros in 2012.[6]

A presentation-oriented way of publishing information on the web leaves room for improvement as it is only partially suited for machine interpretation. In other words, it is very hard for machines to find out what a page is *about* by looking at its structure. In many ways this issue is similar to the problems a visually impaired person experiences when browsing the web (Bechhofer et al., 2006). Considering that most webpages are generated from a structured data source it is at least odd that search engines and other crawlers have to extract meaning in this very roundabout way. Why not present information on the web in a way which *is* directly accessible to machines?

## 3.2 Groundwork

The Semantic Web (Berners-Lee et al., 2001) is a non-intrusive extension of the current web and adds different 'layers' of machine processable semantics. The extension is non-intrusive because each layer builds on another layer without modifying it, see Figure 3.1. Semantics on the web is organised using the same philosophy that underlies current web technologies: transparent heterogeneous networks of linked resources, similar to the Semantic Nets of Section 2.2.3. The first two layers of this cake introduce standards for *character encoding* and *identification*, and a modular mechanism for describing the *structure* of data.

[6] "Europe's Search Engine Marketing Investment Exceeds EUR 8 Billion In 2012", by Mary Beth Kemp, `http://www.forrester.com/Research/Document/Excerpt/0,7211,42707,00.html`

### Character Encoding

Characters should be encoded in a uniform way in order for strings to be interpreted the same way across the globe. This holds for presentation-oriented mechanisms – e.g. browsers should be able to display both chinese and western characters – and extensive data manipulation mechanisms alike. Unicode[7] is a standard adopted by industry that enables computers to consistently represent and manipulate text expressed using a wide range of writing systems. Amongst others, it defines several mappings from (subsets of) some 100k characters onto various encodings using the *Unicode Transformation Format* (UTF). The most well-known of these is UTF-8, an 8 bit encoding with variable-width that is backwards compatible with the older ASCII format.[8]

### Identification

By adopting *Universal Resource Identifiers* (URIs)[9] as means for unique identification of resources on the web, anyone can make statements about other statements similar to the way in which one can now hyperlink from one HTML page to another using a Universal Resource *Location* (URL). In fact, every URL is also a URI (but not vice versa):[10] a URL is a URI used as a location. Alternatively a URI can be used as a *Universal Resource Name* (URN).[11] Note that URI's, including URN's, attach a meaningful machine interpretable *identifier* to resources on the web. They are explicitly not intended to be used as human readable *names*. URI's are structured as follows:

URI = scheme ":" hier-part [ "?" query ] [ "#" profile ]

### Namespaces

A resource on the web is often specified 'within' a *namespace*. A namespace is essentially a collection of URI's in which every resource name is locally unique. The URI of a namespaced resource is composed of a namespace name, and a local name. Usually, the profile part of a URI is used to contain the local name and the composition of scheme and hier-part is used to denote the namespace name. This mechanism can be used to provide *authority* to a namespace, as namespace names can correspond to domain names on the internet. This is a loosely coupled form of 'trust', as the only way to enforce this authority is by dereference-ability of URI's to a location. Namespaces can be abbreviated using so-called *namespace prefixes*, which typically abbreviate the scheme and hier-part to an intelligible shorthand.

[7] See `http://www.unicode.org/`

[8] ASCII: American Standard Code for Information Interchange

[9] The URI specification is defined in RFC 3986, see `http://gbiv.com/protocols/uri/rfc/rfc3986.html`

[10] This simple relation has been the source of much confusion, as many treat URI's as URL's, i.e. by dereferencing from a URI to a URL.

[11] It is often thought that URN's should always use the 'urn' scheme, where syntactic parts of the name are delimited by colons. In fact, *any* URI can be interpreted as either a name or a location, it is the interpretation that turns a URI into a URN or a URL respectively. However, URN's specified using a 'urn' scheme cannot be dereferenced to a URL without a defined mapping.

### Structured Data

The *Extensible Markup Language* (XML)[12] is a flexible way to explicitly encode data in a nested datastructure. It is a subset of SGML,[13] an industry standard meta language for defining exchangeable machine processable document formats. XML diverges from SGML in general by its orientation towards the exchange of *data* on the *web*, and was designed to be compatible with HTML, e.g. XHTML 1.0 Strict is an XML version of HTML and supersedes older versions. Extensibility of XML is provided by the XML Schema[14] language that can be used to describe restrictions on the structure of XML encoded data within a particular namespace; XML Schema is itself an XML dialect. It provides a fixed set of datatypes, such as integers, strings, date, that can be used by XML parsers when interpreting a document.

Important to note, however, is the fact that although the *structure* of some data can be expressed in XML according to some schema, the way in which this data should be interpreted, i.e. its *semantics*, is externally defined (if at all). Even when we *do* commit to a particular interpretation of the nested structure of XML, the least we can say is that it does not convey very expressive semantics.

## 3.3 Lightweight Semantics

"A little semantics goes a long way"

*James Hendler*

The first step towards the Semantic Web is the specification of a way to add simple, relatively lightweight metadata to documents on the web. Currently, the de facto metadata representation languages on the semantic web are the Resource Description Framework (RDF) and the RDF Schema vocabulary description language extension to RDF. Contrary to other knowledge structuring languages, such as Topic Maps (Biezunski et al., 1999)[15] or even UML, which have XML syntaxes and can thus be used to publish metadata on the web, RDF and RDF Schema are *web languages*. That is, the structure of the RDF language is designed to mimic the way in which information is stored on the web. This way, metadata can be distributed across multiple documents or locations; and is extensible as any RDF resource can point directly to other RDF resources in the same way that HTML documents can.

RDF is often criticised as being too technical, having unwieldy semantics and verbose syntax, and at the same time providing only a limited vocabulary of primitives for (lexical) knowledge organisation. These attributes would

[12] See `http://www.w3.org/TR/xml/`

[13] SGML: Standard Generalized Markup Language

[14] See `http://www.w3.org/TR/xmlschema-1/` (structure) and `http://www.w3.org/TR/xmlschema-2/` (datatypes)

[15] See `http://www.topicmaps.org/`, the XML syntax for Topic Maps is XTM, `http://www.topicmaps.org/xtm/index.html`

make it a less likely candidate for intuitive knowledge representation. In part, this can be overcome because RDF is in general more expressive than Topic Maps – RDF descriptions have a higher granularity – and the semantics of both languages can be mapped to each other (Pepper et al., 2006).[16] A related initiative is the Simple Knowledge Organisation System (SKOS) described in Miles and Bechhofer (2008) that defines an RDF data model for expressing the taxonomies, classification structures and thesauri used in many applications to structure and organise knowledge and information such as WordNet.[17]

In fact, there is a subtle distinction between the purpose of languages such as Topic Maps and SKOS, and that of RDF. The former are designed to describe what documents and other information sources (such as database entries) are *about*. Ultimately, these languages are designed to describe indexes on information sources akin to the old fashioned card indexes used in libraries. Both SKOS and Topic Maps are typically used to capture lexical categories, *terms*, rather than semantic categories, *concepts*. In fact, the SKOS specification is very explicit about its purpose and states: "SKOS is not a knowledge representation language" (Miles and Bechhofer, 2008). Although RDF is similarly designed to state facts about information sources, these statements are not necessarily intended to capture (only) the information content of a source: it can be used to express *more*.

### 3.3.1 RDF: The Resource Description Framework

RDF is a lightweight and flexible way to represent metadata on the web. It is a simple assertional language that defines a way to make statements *about* things on the web. These statements take the form of subject, predicate, object triples $< s, p, o >$ a syntactic variant of traditional binary predicates, e.g. $p(s, o)$. The assertion of such a triple is defined to mean that predicate $p$ is a relation between $s$ and $o$. Each part of the triple, i.e. each RDF name, denotes a *resource*.

How a name is treated depends on its syntactic form: URI references are treated as logical constants, but plain *literals* of the form "literal value" denote themselves and have a fixed meaning. A literal that is typed by an XML Schema datatype, e.g. "1"^^ xsd:Integer denotes the value resulting from a mapping of the literal value (the enclosed character string) by the datatype. For instance, the literal "1"^^ xsd:Integer denotes the integer value 1, whereas "1" simply denotes the exact string of characters "1", including the quotes. A resource that has a name which is a URI reference, denotes the entity that can be identified by means of the URI. It *does not* denote the URI itself; nor does it necessarily denote the entity found at the location when the URI is dereferenced as if it were a URL. In other words, a RDF resource can be *anything*, and does not have to exist on the web. Furthermore, a URI cannot be used to identify multiple entities.

A collection of interconnected RDF triples constitutes a directed *graph*, where nodes are the subjects and objects of assertions, and properties are edges. Figure 3.2 shows an example graph that expresses the phrase "Joost Breuker supervises Rinke Hoekstra". In this example, the names "Joost Breuker" and

[16] See e.g. the working group note of the RDF/Topic Maps Interoperability Task Force (RDFTM) at `http://www.w3.org/TR/rdftm-survey/` for an overview.

[17] See `http://www.w3.org/2004/02/skos/`. The SKOS data model itself is expressed in OWL Full.

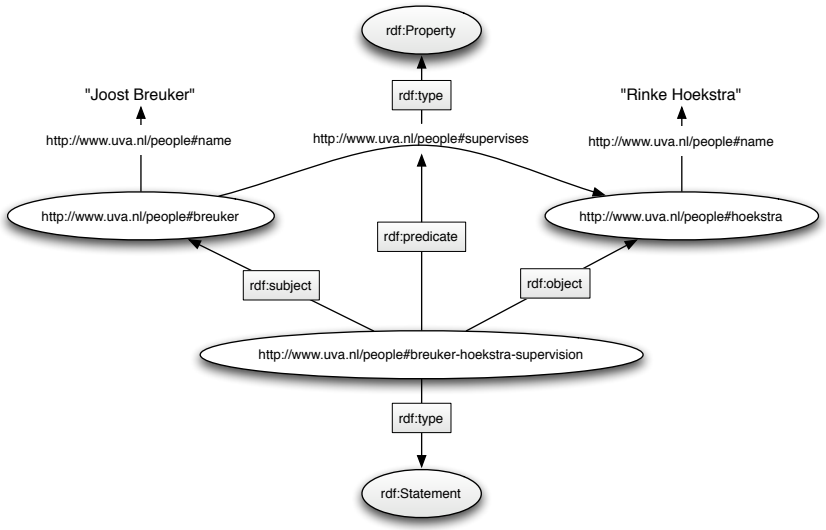


Figure 3.2: An RDF graph representation of the phrase "Joost Breuker supervises Rinke Hoekstra"

"Rinke Hoekstra" are literal values, connected to resources that *are* the respective persons. These resources are identified by URI's in the `http://www.uva.nl/people` namespace: the URI `http://www.uva.nl/people#breuker` denotes Joost Breuker, the actual person. In the following I will abbreviate URIs in the `http://www.uva.nl/people` namespace using the prefix 'uva'.

The supervision relation is denoted by the predicate uva:supervises that holds between the two persons. Not only can this predicate itself be addressed as a resource, as in uva:supervises rdf:type rdf:Property, but any triple as a whole can be *reified*, and explicitly named. In RDF, reification is expressed using the rdf:Statement construct. A resource of type rdf:Statement can explicitly refer to the subject, predicate and object of some property relation using the rdf:subject, rdf:predicate and rdf:object properties, respectively. For instance, the resource uva:breuker-hoekstra-supervision in Figure 3.2 reifies the uva:supervises relation by explicitly pointing to its relata. Note, however, that although the existence of a relation indicates the existence of its reification, an rdf:Statement by itself does not mean that the corresponding relation holds as well (see also Section 7.3.3). In the current example, the assertion of the uva:breuker-hoekstra-supervision statement and its relata does allow us to infer the triple uva:breuker uva:supervises uva:hoekstra.

RDF graphs can be stored in different formats. Most commonly this will be some plain text file, which means that the graph needs to be *serialised*, i.e. 'flattened' or deflated to fit the a sequential order of characters in a file. RDF serialisations are order independent because the order in which triples are added to a file depends on an arbitrary choice for which node is serialised first. There exist three official serialisation syntaxes for RDF: RDF/XML, N-Triple and more recently the Terse RDF Triple Language (Turtle).[18]

[18]See `http://www.w3.org/TR/rdf-syntax-grammar/`, `http://www.w3.org/`

The RDF/XML syntax has the advantage of being a native XML format, though parsing it can be hard as the native order-dependent tree model of XML documents gets in the way of the RDF graph model. Secondly, this makes the syntax notoriously verbose. For instance, the RDF graph of Figure 3.2 is serialised in RDF/XML as follows:[19]

```
<?xml version="1.0"?>
<rdf:RDF
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:uva="http://www.uva.nl/people#"
      xml:base="http://www.uva.nl/people">
   <rdf:Property rdf:ID="supervises"/>
   <rdf:Property rdf:ID="name"/>
   <rdf:Description rdf:about="#breuker" >
      <uva:supervises rdf:resource="#hoekstra"/>
      <uva:name>"Joost Breuker"</uva:name>
   </rdf:Description>
   <rdf:Description rdf:about="#hoekstra" >
      <uva:name>"Rinke Hoekstra"</uva:name>
   </rdf:Description>
   <rdf:Statement rdf:ID="breuker-hoekstra-supervision">
      <rdf:subject rdf:resource="#breuker"/>
      <rdf:predicate rdf:resource="#supervises"/>
      <rdf:object rdf:resource="#hoekstra"/>
   </rdf:Statement>
</rdf:RDF>
```

RDF/XML uses the rdf:about property to state that some rdf:Description concerns the resource indicated by the URI reference. The rdf:resource property connects the predicate of a relation to its object, e.g. the object of the rdf:subject relation in the statement uva:breuker- hoekstra- supervision is the resource uva: breuker.[20] Provided that the type of some resource is known, as is the case with the uva:supervises and uva:name properties, we can directly state the properties of that resource under an element of its type and a rdf:ID attribute that gives the resource's URI. For instance, the serialisation above states that the uva:supervises resource is of type rdf:Property. An equivalent serialisation that does not explicitly introduce the resource, but rather adds information *about* it, is:

---

`TR/2004/REC-rdf-testcases-20040210/#ntriples` and `http://www.w3.org/TeamSubmission/turtle/`, respectively.

[19] Note that the xml:base attribute is used to abbreviate some of the URIs.

[20] Namespace abbreviation can be a bit confusing because of the intermixed use of RDF URI references and xml:ID datatypes for the identifiers. The rdf:ID attribute is an xml:ID and its value is automatically interpreted by an XML parser as an identifier within the base (or default) namespace. This means that an RDF parser does not have to deal with whether an rdf:ID is a full URI or just the fragment, as it will be presented as a full URI of the form *<namespace>+#+<local name>*. The rdf:about and rdf:resource properties, on the other hand, do not specify the identity of the XML element they are located at, but rather *point towards* some other resource. If the URI reference is not fully specified, an RDF parser will have to resolve the URI, and it does this by simply concatenating the value of the xml:base attribute and the URI fragment identifier: *xml:base+<fragment identifier>*. Hence the necessity to prefix the name with a pound character (#).

```
<rdf:Description rdf:about="#supervises">
  <rdf:type
    rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
</rdf:Description>
```

Turtle was developed as a subset of the N3 language,[21] and has a much more readable syntax than RDF/XML. It is similar to N-Triple, in which all triples in the graph are spelled-out, but it allows several shorthand notations. For instance, the omission of the object of a triple in several consecutive statements that are separated by semicolons, and the reserved word a that replaces rdf:type. The Turtle serialisation of the RDF graph in Figure 3.2 is as follows:

```
@prefix uva: <http://www.uva.nl/people#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .


uva:breuker
    uva:name "\"Joost Breuker\"" ;
    uva:supervises uva:hoekstra .

uva:breuker-hoekstra-supervision
    a rdf:Statement ;
    rdf:object uva:hoekstra ;
    rdf:predicate uva:supervises ;
    rdf:subject uva:breuker .

uva:hoekstra
    uva:name "\"Rinke Hoekstra\"" .

uva:name
    a rdf:Property .

uva:supervises
    a rdf:Property .
```

### 3.3.2 The RDF Schema Vocabulary Description Language

Although RDF can already be used to describe quite complex graph structures, it provides little in the way of semantics and automatic inferencing. For instance, there is no standard way to say that both uva:breuker and uva:hoekstra are persons, or that the uva:supervises relation is a relation only between persons. RDF Schema (RDFS) extends RDF with basic primitives that do allow us to express such more generic knowledge.

In fact, most of these primitives are already commonplace in the semantic networks of the 1970s. RDFS introduces the notion of classes (rdfs:Class) and transitive subclass relations (rdfs:subClassOf) which allow to describe taxonomies. Individual resources can be said to belong to a class using the rdf:type property. Under RDFS semantics, the type of a resource is inherited over the rdfs:subClassOf relation: resources belonging to a class belong to all its super

[21] See `http://www.w3.org/DesignIssues/Notation3.html`. N3 allows for several non-RDF constructs such as a forAll loop, rules and paths.

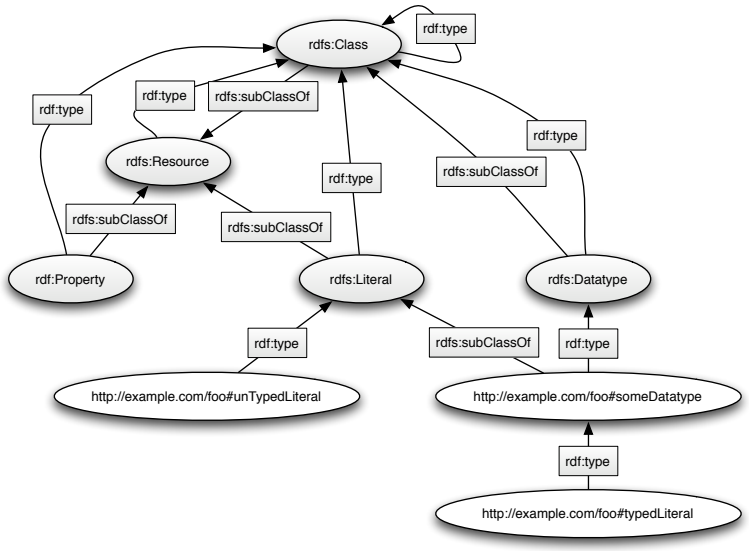


Figure 3.3: Overview of relations between RDFS classes.

classes. The resource rdfs:Class is defined as the set of all classes: all classes in RDFS are of the type rdfs:Class. The class rdfs:Resource is the set of all resources, and is therefore by default the super class of all resources of type rdfs:Class. All RDF literals are members of the class rdfs:Literal. Typed literals are consequently instances of the subclass of rdfs:Literal defined by their datatype. Each XML Schema datatype supported by RDF is a member of the class rdfs:Datatype. The rdf:Property, although still in the RDF namespace, is defined as the class of all properties. The relations between the various RDFS classes are shown in Figure 3.3.

The transitive rdfs:subPropertyOf property can be used to specify that two resources related by one property (the sub property) are also related by another property (the super property). In RDF Schema, any resource of the type rdf:Property can be assigned a domain and range. Domain and range specifications are inherited over the rdfs:subPropertyOf relation. If any two resources are related by a rdfs:domain or rdfs:range property, then the subject of that relation is an instance of rdf:Property and the object an instance of rdfs:Class. If that property then is used to relate two other resources, their type is inferred to be of the classes specified by the domain and range. Multiple domain and range restrictions are interpreted as intersections, i.e. class membership is inferred for all classes in the domain or range, respectively.[22]

To given an example, given just the following three triples:

[22]The rdf:type property is a special property and holds between instances and classes, instead of only classes. It therefore does not have a rdfs:domain property

```
uva:supervises
    rdfs:domain uva:Person ;
    rdfs:range uva:Person .

uva:breuker
    uva:supervises uva:hoekstra .
```

we can infer that uva:supervises is a property, uva:Person is a class and that uva:breuker and uva:hoekstra both belong to that class:

```
uva:supervises
    a rdf:Property .

uva:Person
    a rdfs:Class .

uva:breuker
    a uva:Person .

uva:hoekstra
    a uva:Person .
```

The properties rdfs:label and rdfs:comment can be used to annotate resources in human readable form, using multiple languages (by means of language tags). Values of an rdfs:label property are intended to give a human readable name alternative to the URI of a resource. The range of these properties is the xsd:string datatype:

```
uva:name
    a rdf:Property;
    rdfs:label "Name"@en;
    rdfs:label "Nom"@fr;
    rdfs:label "Name"@de;
    rdfs:label "Naam"@nl;
    rdfs:comment "This property is used to specify a person's name."@en;
rdfs:comment "Dit property wordt gebruikt voor de naam van een persoon."@nl.
```

#### Meta-Modelling in RDFS

RDFS has a non-fixed meta modelling architecture; it can have an infinite number of class layers because rdfs:Resource is both an instance and a super class of rdfs:Class, which makes rdfs:Resource a member of its own subset (Nejdl et al., 2000). All classes (including rdfs:Class itself) are instances of rdfs:Class, and every class is the set of its instances. There is no restriction on defining sub classes of rdfs:Class itself, nor on defining sub classes of instances of instances of rdfs:Class and so on. This is problematic as it leaves the door open to class definitions that lead to Russell's paradox (Pan and Horrocks, 2002). The Russell paradox follows from a comprehension principle built in early versions of set theory (Horrocks et al., 2003). This principle stated that a set can be constructed of the things that satisfy a formula with one free variable. In fact, it introduces the possibility of a set of all things that do not belong to itself: $\{x|x \notin x\}$ given the formula $x \notin x$. We cannot determine whether the class $x$ is a member of

itself: this only holds if and only if it is not a member of itself.

Meta modelling of the type sanctioned by RDFS provides no means to exclude the definition of such classes from extensions of the RDFS semantics. For instance, a straightforward semantic extension of RDFS that allows qualified cardinality restrictions (as the DL's $\mathcal{SHIQ}(D)$ and $\mathcal{SHROIQ}(\mathcal{D})$ do, cf. Section 4.2 and Section 3.5.2, respectively) would allow us to define a class as an instance of itself, or even add a cardinality constraint of 0 on the rdf:type property (Pan and Horrocks, 2003).

In RDFS, the reserved properties rdfs:subClassOf, rdf:type, rdfs:domain and rdfs:range are used to define both the other RDFS modelling primitives themselves and the models expressed using these primitives. In other words, there is no distinction between the meta-level and the domain. In terms of Figure 2.5 in Section 2.3.1, this means that RDFS conflates the language and representation columns.

One example is the circular definition of rdfs:subClassOf and rdfs:Class; rdfs:Class is defined as the rdfs:subClassOf rdfs:Resource, but rdfs:subClassOf is defined as a relation between resources of the type rdfs:Class. Some of the promiscuity of RDFS primitives was resolved by specifying a model theoretic semantics (Hayes, 2004, RDFS MT).[23] However, RDFS MT included the inherent problems with a non-fixed meta modelling architecture (Pan and Horrocks, 2003; Motik, 2007).

In a nutshell, RDFS has its appeals as a relatively lightweight means to add additional semantics to RDF models, but it falls short in many ways as a language for specifying knowledge on the web. As knowledge representation language the RDFS is still quite weak. Other than domain and range restrictions it has no means to describe the requirements for class-membership needed for concise definitions, such as necessary & sufficient conditions, cardinality constraints and negation. For instance, it can be useful to express that two classes are *disjoint*, e.g. to distinguish between animals and plants: no animal can be a plant and vice versa;[24] or to state that every person has at most one father and one mother. It furthermore embraces the 'anything goes' mentality of RDF, which leads to conceptual, computational and decidability issues.

## 3.4 Requirements for Web-Based Knowledge Representation

The ideas behind the Semantic Web initiative was met with a fair amount of enthusiasm by the knowledge representation and acquisition community. The ideal of knowledge sharing that was so prominent in the literature of the early nineties (Neches et al., 1991; Gruber, 1993; Breuker and Van De Velde, 1994, etc.) was not feasible when confined to the traditional libraries or knowledge servers, but the web, and the semantic web, seemed its perfect partner. Knowledge sharing need no longer be controlled through publication in a library, but can be *unattended* and made possible simply by the availability of a knowledge representation at some URL. One of the first proposals for a web-based ontology language – coinciding with the development of RDF and RDFS – was the

[23] See Section 2.5.1 for a description of model theory.

[24] Although some people may turn into vegetables.

Simple HTML Ontology Extensions language, SHOE for short (Heflin et al., 1999)). SHOE allowed the semantic annotation of webpages, had an XML syntax and was frame-based, i.e. concepts are defined as frames with slots (cf. Section 2.2.4). However, like RDF Schema, it did not have a well-defined semantics.

An important requirement was that a web-based knowledge representation language should sanction standard inference. Reasoners complying with the standard should produce the same inferences, given the same input. This is a significant difference with highly expressive languages such as the Knowledge Interchange Format (Genesereth and Fikes, 1992, KIF), which itself did not have reasoners but instead relied on a translation to less expressive languages for which implementations were available. The web ontology language was to be more than just a *specification* language that can be used to exchange formal theories between systems, but rather a *representation* language in its own right. This requirement gains an extra edge as knowledge published on the web is not merely intended for human consumption, but also, and more importantly, for direct communication between different systems that act as knowledge components in a larger, distributed knowledge based system. Only systems that commit to the same representation will be able to properly interpret each others messages.

Information exchange between web-based systems becomes increasingly more mission critical: knowledge-based reasoning should not only be suitably efficient, but sound and complete as well (see Section 2.5.1). A web-based knowledge representation language is therefore subject to the *restricted language thesis* of Levesque and Brachman (1987) and needs to trade off expressive power with computational efficiency. The relaxed view of Doyle and Patil (1991) may well hold for applications where the user is in direct control, and understands the rationale of the system. But it is untenable on the web, where any user may use any knowledge representation for any purpose in any system. In other words, because standard reasoning should be efficient and decidable, the language should be limited in computational complexity and expressiveness.

From the outset, it was clear that the language should strike a fine balance between the open nature and expansiveness of the web on the one hand, and formal, well-defined semantics on the other. It should therefore build on top of existing (semantic) web standards such as XML, RDF, and RDF Schema (cf. Figure 3.1), and allow knowledge engineers and users to freely extend and reuse the knowledge made available on the web. This meant not only that ontology files should be able to *import* each other from URL's, but also that the semantics of every ontology should take into account that it can be *imported* and extended. In other words, the ontology language should assume that any ontology represents incomplete knowledge and adopt the *open world assumption*; a reasoner should only make inferences based on those statements that are known to be true. This is in stark contrast with common practice in many rule based systems, where inference generally takes place on the assumption that any statement not known to be true is false. In a closed system this closed world assumption (or negation as failure principle) is quite sensible, but it can be quite dangerous on the web. For instance, it would mean that what is inferred for an individual given the axioms in an ontology $A$ may be inconsistent with that which is inferred for the same individual when imported to another

ontology $B$. In the former case, any inference based on the axioms in $B$ is assumed not to hold, where they do hold in the latter case.

The DAML+OIL language of Connolly et al. (2001)[25] was a proposal for a semantic web knowledge representation language in the tradition of the earlier KRSS (Patel-Schneider and Swartout, 1993, see Section 4.2). It had a DL-style semantics and was a combination of two languages; the DARPA Agent Markup Language (DAML-ONT),[26] and the Ontology Inference Layer (Fensel et al., 2000, OIL).[27]

Both approaches were very similar, they were:

- languages with formal semantics,
- extensions of XML and RDF(S),
- intended for defining concepts and relations,
- to be shared on the *web*

DAML+OIL was the first knowledge representation language that combined the principle of knowledge sharing with the formal semantics and limited expressiveness of DL, allowing for sound and complete automated reasoning (Baader et al., 2003). Where the semantics of OIL relied on a translation to the description logic $\mathcal{SHIQ}(d)$ (Horrocks, 2000) and was not necessarily compatible with RDF, DAML+OIL was rather a syntactic variant of a description logic and aimed to maximise compatibility with RDFS. It was the first proper knowledge representation language specifically tailored for the web.

## 3.5 The Web Ontology Language

OWL extends the syntax and semantics of RDFS with a fair number of constructs and combines the description logics semantics of DAML+OIL with more rigorous semantic and syntactic compatibility regarding RDF and RDFS. It furthermore prescribes how ontologies should be published on the web, and incorporates a mechanism akin to that of SHOE for ontology imports. In the following I give a brief, non exhaustive, overview of the OWL vocabulary and its semantics. The interested reader may refer to Horrocks et al. (2003); Bechhofer et al. (2004) for a full overview.

### 3.5.1 OWL

The various requirements for a web ontology language led to the specification of three *species*, each emphasising different language features: OWL *Full*, *DL*, and *Lite*. OWL Full is a semantic extension of RDFS that adds a number of knowledge representation primitives that were previously unavailable. It emphasises expressiveness and compatibility with RDFS: the RDFS interpretation of an OWL Full ontology is consistent with its OWL Full entailments; and any

[25] See `http://www.w3.org/TR/daml+oil-reference`

[26] Research funded by *Darpa*, the Defense Advanced Research Projects Agency, see `http://www.darpa.gov` and `http://www.daml.org`

[27] Developed within the EU funded OnToKnowledge project, see `http://www.ontoknowledge.org/oil`

OWL Full ontology is valid RDFS and vice versa. This means, amongst others, that in OWL Full the primitives of the language (i.e. the meta logical symbols) are part of the *domain*. However, as proved by Motik (2007), this mixing of logical and metalogical symbols in OWL Full leads to undecidability (see Section 3.3.2).

The second species, OWL DL, is a syntactic subset of OWL Full, but limits the semantics to the $\mathcal{SHOIN}(D)$ description logic (see Table 2.4). At the time this language was the maximally expressive, decidable description logic for which efficient algorithms were known and use cases existed. To retain decidability, the compatibility with RDFS is restricted to a specific subset of that language. This means that although every OWL DL ontology is valid OWL Full and thus RDFS, this does not hold the other way around. However, because any conclusion given the OWL DL semantics is a valid conclusion in OWL Full, the RDFS interpretation of an OWL DL ontology is consistent with its DL entailments.

Because $\mathcal{SHOIN}(D)$ is very expressive, it allows the construction of exceedingly complex expressions which can be quite hard to grasp for an ontology engineer. Furthermore, reasoning with this logic is computationally expensive and intractable. The third species, OWL Lite, is meant to alleviate some of the problems of its bigger brother. It is a syntactic subset of OWL DL, where the semantics is designed to be limited to the $\mathcal{SHIF}(D)$ description logic. Similar to OWL DL, any OWL Lite conclusion is a valid OWL DL conclusion, and the RDFS interpretation of an OWL Lite ontology is therefore consistent with its DL entailments. Unfortunately it soon turned out that the syntactic limitations imposed on OWL Lite did not, in fact, limit it to $\mathcal{SHIF}(D)$, and most of the semantics of OWL DL can be expressed using elaborate combinations of OWL Lite constructs.

OWL follows the distinctions of DL between *class* axioms, *property* axioms and *individual* assertions. Figure 2.11 shows how the OWL DL constructs owl:Class and owl:Individual map onto their corresponding DL constructs.

**Class Axioms** Since not all RDFS classes are valid OWL DL and OWL Lite classes, OWL introduces the owl:Class which is the rdfs:subClassOf rdfs:Class consisting of all valid OWL classes. Because in OWL Full all RDFS classes are valid, owl:Class is equivalent to rdfs:Class under OWL Full semantics.

Just as in RDFS, owl:Classes can be related using rdfs:subClassOf properties. All OWL classes are a subclass of the class owl:Thing – which in OWL DL is equivalent to $\top$. The class owl:Nothing represents the empty class – in OWL DL equivalent to $\bot$ – and is defined as the complement of :owl:Thing. In OWL Full owl:Thing is equivalent to rdfs:Resource.

Two classes that are the rdfs:subClassOf each other, are *equivalent*, which can be directly expressed using the owl:equivalentClass property. An owl:Class is either *named* or *anonymous*. A named class is defined using *class axioms* that restrict it to be disjoint with, equivalent to or the subclass of one or more other classes. Disjointness of two classes means that the intersection of both is the empty set. There are three types of anonymous classes:

- *Nominals* are defined by exhaustively enumerating all individual class members, for instance:[28]

[28]The _:x in the example is a *blank node*, an RDF resource without a URI. This blank node can

```
:Animal a owl:Class ;
    owl:equivalentClass _:x ;

_:x owl:oneOf (:fluffy :felix :bertha :snuffy :max :babel) .
```

states that the class :Animal is equivalent to the set consisting of the individuals :fluffy, :felix, :bertha, :snuffy, :max and :babel. In DL-style syntax:

$$\text{Animal} \equiv \{\mathit{fluffy}, \mathit{felix}, \mathit{bertha}, \mathit{snuffy}, \mathit{max}, \mathit{babel}\}$$

- *Operator* classes are defined using the standard set theoretic operators union, intersection and complement, for instance:

```
:Animal a owl:Class ;
  owl:equivalentClass [
    owl:unionOf (:Mammal :Fish :Reptile :Insect)] .
```

states that the class :Animal is equivalent to the union of the classes :Mammal, :Fish, :Reptile and :Insect. In DL-style syntax:

$$\text{Animal} \equiv \text{Mammal} \sqcup \text{Fish} \sqcup \text{Reptile} \sqcup \text{Insect}$$

- *Restriction* classes are defined as the set of all individuals that satisfy a restriction on a property. The restriction can be *existential* or *universal*, i.e. it expresses that some or all values of a property at the restricted class must be a member of some other (anonymous) class. Or, it may restrict the *cardinality* or prescribe a specific individual *value* of the property. For instance:

```
:Animal a owl:Class ;
  rdfs:subClassOf [
    a owl:Restriction ;
    owl:onProperty :cell_type ;
    owl:allValuesFrom :Eukaryote ] ;

  rdfs:subClassOf [
    a owl:Restriction ;
    owl:onProperty :cell_type ;
    owl:minCardinality 1 ] ;

  rdfs:subClassOf [
    a owl:Restriction ;
    owl:onProperty :friend ;
    owl:hasValue :joe ] .
```

states that the class :Animal is the subclass of all things that have only Eukaryotic cells, have at least one value for the cell type property, and are the individual :joe's friend. Multiple rdfs:subClassOf statements on one

be left implicit in the Turtle syntax by placing the triples in which the blank node occurs between square brackets (see the operator class example). The space separated list of resources between parentheses is the Turtle abbreviation for an rdf:List.

class are, as in RDFS, interpreted as the *intersection* of the super classes. In DL-style syntax:[29]

$$\begin{aligned} \text{Animal} &\sqsubseteq \text{cellType } \textbf{only } \text{Eukaryote} \\ &\sqsubseteq \text{cellType } \textbf{min } 1 \\ &\sqsubseteq \text{friend } \textbf{value } \textit{joe} \end{aligned}$$

What can be confusing is that OWL allows one to state many things in different, but equivalent ways. For instance, note that the owl:minCardinality restriction in the definition of Animal is equivalent to:

$$\text{Animal} \sqsubseteq \text{cellType } \textbf{some } \text{owl:Thing}$$

Also, the owl:allValuesFrom restriction is trivially satisfied if an individual has no value for the cell_type property.

**Individuals** Individuals can be said to belong to a class in the usual RDFS way, i.e. by asserting an rdf:type triple between the individual and class. Asserting property values on individuals is similarly straightforward, e.g. to state that :joe is a :friend of :babel the fish, we write:

```
:babel a :Fish ;
    :friend :joe .
```

Or, in DL syntax:[30]

$$\begin{aligned} \textit{babel} &\in \text{Fish} \\ \text{friend}(\textit{babel}, \textit{joe}) & \end{aligned}$$

As mentioned earlier, individuals in OWL are not subject to the unique name assumption. We have seen that two individual names can be inferred to have the same model through the use of a functional property. The same can also be achieved in a less round-about way by asserting an owl:sameAs relation between two individuals. The other way around, the owl:differentFrom property can be used to assert that two individuals do not have the same model.

Remember that in description logics individuals are elements and classes are subsets of the domain $\Delta^{\mathcal{I}}$. As a consequence, OWL DL only allows individuals in the subject and object position of property relations, because otherwise subsets of the domain are themselves elements in the domain; which leads to undecidability (Motik, 2007).

**Property Axioms** Another important extension of the RDFS vocabulary is the addition of property axioms beyond the rdfs:domain and rdfs:range attributes:

[29] The examples in this chapter and chapters 5,7 do not use the standard DL-style syntax, but a more readable form that is akin to the syntax used by common OWL editors such as Protégé and TopBraid Composer.

[30] Since the RDF syntax for OWL is quite verbose – even when using Turtle – the following uses the DL-style notation for OWL constructs.

- Any property can be stated to be the owl:inverseOf another property. Any two individuals related via a property in one direction, are related by the inverse of that property in the other direction. As a consequence, the domain of a property is equivalent to the range of its inverse and vice versa.
- An OWL property can be stated to be the rdfs:subPropertyOf another property. The set of models of that property is a subset of the models of its super property. Consequently, the domain and range of the property are the respective subclasses of the domain and range of the super property.
- *Functional* properties have exactly one individual as range; this means that any individuals asserted in the object position of that property have the same model, and are thus *the same*. This construct should not be confused with cardinality restrictions on classes. A functional property expresses a *global* cardinality restriction; but it applies separately to every individual in the knowledge base on which the property is used, regardless of the class it belongs to. For instance, consider the functional father property. Then from

$$\begin{aligned} \mathsf{father}(\mathit{babel}, \mathit{bubba}) \\ \mathsf{father}(\mathit{babel}, \mathit{elvis}) \end{aligned}$$

$$\begin{aligned} \mathsf{father}(\mathit{max}, \mathit{ludwig}) \\ \mathsf{father}(\mathit{max}, \mathit{beethoven}) \end{aligned}$$

  an OWL reasoner will infer that $\mathit{bubba} = \mathit{elvis}$ and $\mathit{ludwig} = \mathit{beethoven}$.
- *Inverse functional* properties express a similar global cardinality restriction but have exactly one individual as *domain*; any individuals asserted in the subject position, for the same object of that property have the same model:

$$\begin{aligned} \mathsf{father\_of}(\mathit{bubba}, \mathit{babel}) \\ \mathsf{father\_of}(\mathit{elvis}, \mathit{babel}) \end{aligned}$$

  gives $\mathit{bubba} = \mathit{elvis}$. An inverse functional property has the same effect as the inverse property of a functional property, but is more concise as it does not require the existence of a functional property.
- *Transitive* properties allow property values to propagate along a chain of connected properties. Examples of transitive properties are taxonomic relations such as the rdfs:subClassOf property and mereological relations such as part_of, e.g.:

$$\begin{aligned} \mathsf{part\_of}(\mathit{piston}, \mathit{engine}) \\ \mathsf{part\_of}(\mathit{engine}, \mathit{car}) \end{aligned}$$

  gives $\mathsf{part\_of}(\mathit{piston}, \mathit{car})$.

- *Symmetric* properties express that any two individuals related via the property in one direction are also related in the converse direction. In other words, a symmetric property is equivalent to its inverse. Given, $\mathsf{sibling}(\textit{eefje}, \textit{suzan})$ the reasoner infers $\mathsf{sibling}(\textit{suzan}, \textit{eefje})$.
- Any two or more properties can be stated to be equivalent to each other. The classes in the domain and range of the properties are correspondingly equivalent, and any pair of individuals related via one property is also related via the other.

The semantics of these property axioms is in fact very powerful because the axioms hold *globally*. A second reason is that where (in DL) class axioms only effect the semantics in the TBox, property axioms in the RBox can have a significant implications for axioms in the the TBox, such as the effects on classes in the domain and range of inverse, equivalent and sub properties.

The domain and range of properties in OWL are in principle not restricted in any way, i.e. properties and restrictions can range over concrete data values such as strings and integers. However, it has been shown that the combination of datatypes and description logics, i.e. the incorporation of a concrete domain into a concept language, may cause undecidability (Lutz, 1999). To retain decidability, OWL DL adopts the solution of Horrocks and Sattler (2001); Baader and Hanschke (1991) who introduce an additional domain for concrete data values $\Delta_D^{\mathcal{I}}$ in $N_{I_c}$ which is disjoint with the domain of abstract individual objects $\Delta^{\mathcal{I}}$ in $N_{I_a}$, where $N_I = N_{I_c} \cup N_{I_a}$. Analogous to classes and individuals in $\Delta^{\mathcal{I}}$, datatypes are interpreted as subsets of $\Delta_D^{\mathcal{I}}$ and data values are elements of $\Delta_D^{\mathcal{I}}$. In practice, this resulted in a distinction between two disjoint types of properties: properties that have only literal values, *datatype* properties or concrete roles in $N_{R_c}$, and properties that have only individuals as values, *object* properties or abstract roles in $N_{R_a}$ where $N_R = N_{R_c} \cup N_{R_a}$. Formally, a datatype property $\mathsf{D}$ is defined as a binary relation, a subset of the set of all object data value pairs: $\mathsf{D}^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta_D^{\mathcal{I}}$. In other words, properties can not have data values in the subject position in OWL DL. Consequently, datatype properties cannot have an inverse, be symmetric, inverse functional or transitive. This restriction does not hold in OWL Full as $\Delta_D^{\mathcal{I}}$ and $\Delta^{\mathcal{I}}$ are not disjoint under Full semantics.

Although generally a property can be of any or all of the property types described, some combinations between property types and class restrictions require extra care or are forbidden in DL (Horrocks et al., 2006). In particular *composite* properties may cause decidability issues in several cases (Horrocks et al., 1999). A composite property is any property whose semantics is defined as a (possible) sequence of (other) properties: i.e. the transitive properties. A *complex* property is a property whose definition is defined in terms of a composite property, i.e. the super properties, equivalent properties and inverse of transitive properties. In DL, complex properties are not allowed in cardinality restrictions, nor are they allowed to be functional or inverse functional.

### 3.5.2 OWL 2

In October 2007, the W3C started a new working group to develop a successor to the OWL Web Ontology Language: OWL 2. This language extends the original OWL with a number of features for which effective reasoning algorithms

are available, and which meet real user needs.[31] Until now, most research on OWL 2 has been directed towards a new version of the description logics dialect of OWL 1,[32] and is based on the $\mathcal{SROIQ}$ description logic described by Horrocks et al. (2006), but extends it with datatypes and punning (see below).

A large part of the development of OWL 2's features has taken place during, and in between OWLED workshops, i.e. outside of the W3C standardisation process.[33] At the start of the working group, efficient implementations of OWL 2 were already available in standard reasoners such as Pellet, FaCT++, Racer and KAON2.[34] Because of its early and wide adoption by both implementers and users, the OWL 2 effort has a good chance of reaching W3C recommendation status.

OWL 2 has a modular and extensible architecture in the form of language profiles, user defined datatypes and semantic annotations. It resolves several ambiguities in the RDF mapping, such as the ability to use owl:oneOf with or without an owl:equivalentClass axiom, and extends the OWL 1 vocabulary with shorthand notations for frequently used combinations of OWL 1 primitives such as disjoint union.

Some of OWL 2's most significant additions lie in the expressiveness for property axioms. *Asymmetric* properties can be used to express e.g. that if something is bigger than something else, this does not hold vice versa, in other words: the property is disjoint with its inverse. The asymmetry of a property can thus also be expressed in terms of two *disjoint properties*. Disjointness between two properties means that their sets of models are disjoint; no pair of individuals can be in both sets. This can be used to express e.g. the disjointness of the brotherhood and sisterhood properties: if Mary is Bob's sister, she is not his brother. A nice side-effect of property disjointness is thus that it can be used to state that some individual is *not* related to another individual via some property. OWL 2 provides syntactic sugar in the form of *negative property assertions* on individuals that can express this without the need to create a dummy disjoint property.

The feature of *property chain inclusions* – also called complex role inclusion – allows the transitive 'inheritance' of some property over a chain of properties Horrocks et al. (2006). For instance,

$$\mathsf{owns} \circ \mathsf{has\_part} \sqsubseteq \mathsf{owns}$$

expresses that if some individual owns an individual, it is also the owner of that individual's parts.[35] As shown in Chapter 7 role inclusions are very powerful primitives. Important to note, however, is that the chain of properties on the left is only a sub property of the property on the right hand side: they are not equivalent. Also, property chains, like transitive properties, are *compound*, which means that their use in combination with other property and class axioms is subject to the same restrictions under OWL 2 DL.

[31] See the WG website at `http://www.w3.org/2007/OWL/`. At the time of this writing, the exact specification of OWL 2 was not determined yet. Please refer to the website for the latest version.

[32] To avoid confusion, from here on I refer to the original OWL standard as OWL 1

[33] OWLED: "OWL: Experiences and Directions". See `http://www.webont.org/owled/`

[34] See `http://pellet.owldl.com`, `http://owl.man.ac.uk/factplusplus/`, `http://www.racer-systems.com` and `http://kaon2.semanticweb.org/` for respective descriptions and downloads.

[35] Where $\circ$ is a concatenation operator.

OWL2 properties can be *reflexive* or *irreflexive*, to express such things as that every thing is part of itself, or that no effect can be its own cause. Global reflexivity of a property makes it apply to *all* individuals. A more subtle way to express reflexivity is *local reflexivity* of properties in class descriptions. This enables us to express that each member of the restricted class is related to itself via that property. A local reflexivity restriction takes the form of a '**self**' restriction on an object property. For instance, the class definition:

$$\mathsf{Narcissist} \equiv \mathsf{Person} \sqcap \mathsf{likes}\ \textbf{some self}$$

states that the class Narcissist is equivalent to the set of persons who like themselves. To indicate the difference, if the likes property were itself reflexive then *every* individual would like itself. The **self**restriction can only be used in combination with an existential restriction.

A second new class axiom is the *qualified cardinality restriction* (QCR). QCR's add the ability to restrict the cardinality of a property to a particular range, e.g. a table has exactly four legs as parts, but can have more parts which are not legs:

$$\mathsf{Table} \sqsubseteq \mathsf{has\_part}\ \textbf{min}\ 4\ \mathsf{Leg} \sqcap \mathsf{has\_part}\ \textbf{max}\ 4\ \mathsf{Leg}$$

#### Punning

As pointed out multiple times, description logics retain decidability by disallowing meta modelling and redefinition of meta logical symbols that are part of the vocabulary of the language itself. In practice, this means that the sets of names for the vocabulary, classes, properties and individuals are mutually disjoint (see Section 2.5.1). OWL 2 drops this requirement, but obtains the same effect by explicitly marking the proper interpretation for each occurrence of a name: names are given a *contextual semantics*, c.f. Motik (2007). This means that names can be treated as *any or all of* these types, e.g. the meaning of a name as class is in no way affected by the meaning of a name as individual. This trick is called *punning*, wordplay. Although the interpretation of a name may vary, the name is still considered to denote the same entity. Punning enables straightforward meta modelling without the impact on decidability that RDF and OWL Full modelling have. Because of interaction with the semantics of OWL Full, punning is not allowed between data and object properties.

#### Publishing and Metadata

The support for metadata and annotation of ontologies in OWL 1 was both under specified and limited. These limitations are largely the result from a lack of experience in how OWL ontologies are used by various users and systems. A few years onwards, this experience is present and allows OWL 2 to embody a more comprehensive view on annotations and metadata.

Firstly, OWL 1 supports the standard rdfs:label and rdfs:comment properties to respectively give meaningful names and attach descriptions to classes, properties and individuals. These properties are interpreted as owl:Annotation Property, a special type of property that does not carry any semantics in DL, but is interpreted as rdf:Property under OWL Full semantics. Users are free to define additional annotation properties, but their use is severely limited in DL.

Annotation properties cannot be used *in* class restrictions, e.g. to define the set of all persons with more than one rdfs:label name. And they cannot be used *on* class restrictions either. Especially this last restriction was considered an unnecessary shortcoming, and OWL 2 DL extends OWL with *axiom annotations*, that is any axiom (such as a class restriction) can be annotated.

Furthermore, OWL 2 extends the annotation system with rdfs:subPropertyOf relations, and the specification of domain and range for annotation properties. Under DL semantics these do not carry semantics, but the extension allows a larger portion of OWL Full ontologies to be valid in OWL 2 DL as well. Ways are currently being investigated to allow *rich* annotations which can be used to attach information to axioms that is interpretable by extensions of OWL reasoners.[36] For instance, PRONTO[37] extends the Pellet reasoner with a probabilistic reasoning engine that uses annotations on rdfs:subClassOf to assert the probability of that relation to hold between two classes.

Another area where OWL 1 was significantly underdeveloped is its means to resolve owl:import relations between ontologies. In OWL 1, the owl:import property is simply defined as a transitive property that holds between two ontologies, i.e. the domain and range of owl:import is owl:Ontology. However, because the specification is silent on *how* ontologies are to be identified on the web, there is no standard means by which the import mechanism can be used. This weak definition was a relatively late addition to the language, resulting from a compromise between the ontology-as-theory and ontology-as-document perspectives. Regardless of its ambiguous definition, it turned out that this owl:import property was widely used. The OWL 2 specification (Motik et al., 2009) is therefore much more explicit in this respect, and recommends a method that combines the imports mechanism with simple versioning. OWL 1 defines several ontology properties for versioning (owl:priorVersion, owl:backwardCompatibleWith and owl:incompatibleWith) but again, these do not have a prescribed normative, nor recommended, effect on the semantics of imports. Because there was no recipe for importing a particular version of an ontology, each OWL tool used its own particular way to deal with versions and perform resolution of ontology URI's to ontology documents.

In a nutshell, the mechanism in OWL 2 works as follows. If an OWL 2 ontology imports another ontology, this is done from the URL dereferenced (in the standard way) by the URI value of the owl:imports property. The imported ontology should have either that URI as the value of its ontology URI (i.e. the URI of the owl:Ontology element), or as the value of its owl:versionInfo property. Because user agents, such as ontology editing tools may specify their own method for resolving URI's to locations, the ontologies need not necessarily be published on the web, but may be accessed from ontology repositories or a local file system. If an ontology is imported by another ontology, then its import closure becomes part of the import closure of the importing ontology. The axiom closure of an ontology is the smallest set that contains all axioms from all ontologies in its import closure.[38] The versioning properties from OWL 1 may be used by user agents to raise a flag when e.g. the value of an owl:incompatibleWith property corresponds with a owl:versionInfo or onto-

[36] See `http://www.w3.org/2007/OWL/wiki/Annotation_System`

[37] See `http://clarkparsia.com/weblog/2007/09/27/introducing-pronto/`

[38] See also definitions 5.4.2 and 5.4.3 in Section 5.4 for a more formal discussion.

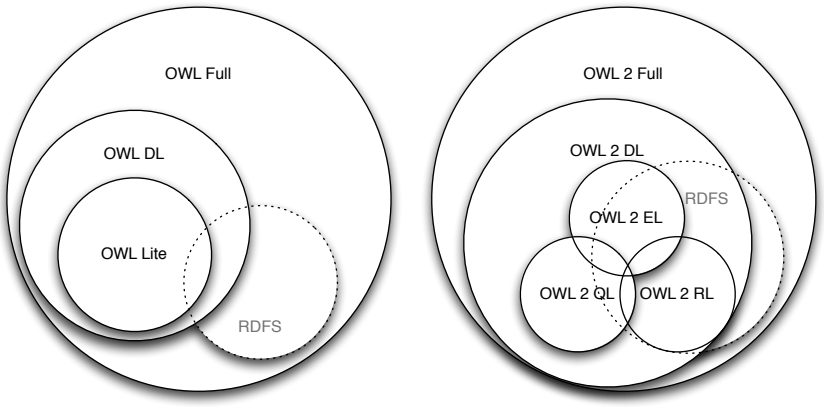


Figure 3.4: OWL 1 species vs. OWL 2 Profiles

logy URI.

#### Profiles

The OWL 2 specification includes a number of so-called *profiles*, some of these are well-known tractable subsets of the OWL 2 DL specification, others are more expressive, but do not have the full semantics of OWL 2 Full (Cuenca Grau et al., 2009). The motivation for providing these profiles is that many existing ontologies tend to use only a particular subset of the language constructs available in DL; and significant increase of reasoner performance can be achieved through reasoning using a less expressive language. It is thought that a standard library of logical profiles with a particularly likeable tradeoff between expressiveness and computational complexity can overcome some of the problems experienced when defining the OWL 1 Lite version. In particular the profiles are:

- restricted by *syntax*. The semantics of a profile's syntax is provided by the OWL 2 DL specification.
- defined by logics that can handle at least some interesting inference service in polynomial time with respect to either:
  - the number of facts in the ontology, or
  - the size of the ontology as a whole.

This section gives a brief overview of the profiles defined in OWL 2 and their typical application areas, see Figure 3.4.[39]

**OWL 2 EL** The EL profile, based on the $\mathcal{EL}++$ DL introduced in Baader et al. (2005), is an extension of the $\mathcal{EL}$ description logic. Its primary strength lies in

[39] For an up-to-date overview, see `http://www.w3.org/2007/OWL/wiki/Profiles`

the ability to reason in polytime on ontologies with large TBoxes, and was designed to cover the expressive power of several existing large-scale ontologies, such as

**SNOMED-CT®**
is a large-scale commercial ontology that defines the international standardised terminology of the IHTSDO,[40] which is used in the health care systems of both the US and the UK. It consists of about 500k named class definitions.[41]

**Gene Ontology**
is an ontology of 25 thousand terms related to genes and gene properties.[42]

**GALEN**
About 95% of this closely interlinked multi-lingual ontology of medical terms is covered by $\mathcal{EL}++$. The GALEN ontology contains about a thousand class definitions.[43]

Where $\mathcal{EL}$ supports *conjunction* and *existential* restrictions, $\mathcal{EL}++$ extends this with nominals and role inclusions. It is lightweight and supports sound and complete reasoning in polynomial time. The most significant difference with OWL 2 DL ($\mathcal{SROIQ}$) is that it drops the owl:allValuesFrom restriction, though it does support rdfs:range restrictions on properties, which can have a similar effect. $\mathcal{EL}++$ is not a profile of OWL 1 DL as it supports the complex role inclusion axioms of OWL 2, and it is more expressive than OWL 1 Lite in that it allows for existential owl:someValuesFrom where OWL 1 Lite doesn't.

**OWL 2 QL** Reasoners developed for OWL DL 1.0 and 1.1 are optimised for reasoning on TBox axioms, and are relatively inefficient when dealing with ontologies that have relatively uncomplicated class definitions, but contain a large number of ABox assertions. The QL profile of OWL 2 was developed to efficiently handle query answering on such ontologies, and adopts technologies form relational database management. It is based on the DL-Lite description logic of Calvanese et al. (2005). By itself DL-Lite is a very restricted a profile of both OWL 2 DL and OWL 1 DL but the QL fragment of OWL 2 extends DL-Lite with more expressive features such as the property inclusion axioms, i.e. owl:subPropertyOf, and functional and inverse-functional object properties of OWL 1. However, it also includes the top and bottom roles of OWL 2, which adds expressiveness beyond that of OWL 1 DL.

**OWL 2 RL** The OWL 2 RL profile is based on so-called Description Logic Programs (Grosof et al., 2003), which is a subset of OWL DL 1.0 and the Horn profile of First Order Logic (FOL) (see Figure 3.5). DLP enables the interaction

[40] International Health Terminology Standards Development Organisation, see `http://www.ihtsdo.org`

[41] Systematized Nomenclature of Medicine, Clinical Terms, see `http://www.snomed.org`

[42] See `http://www.geneontology.org/`

[43] Generalised Architecture for Languages, Encyclopaedias and Nomenclatures in medicine, see `http://www.openclinical.org/prj_galen.html`

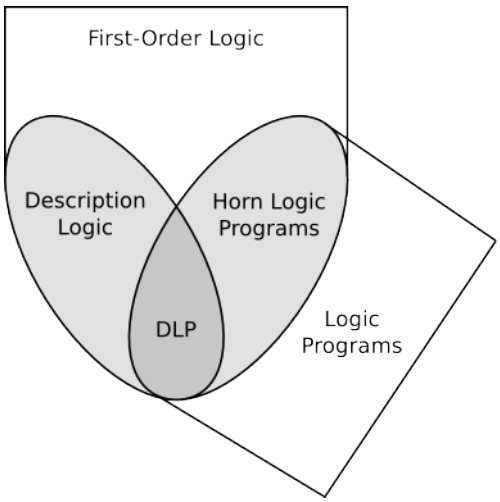


Figure 3.5: The DLP profile, adapted from Grosof et al. (2003)

between DL and rules[44], and effectively allows knowledge engineers to build rules 'on top' of ontologies and vice versa. DLP axioms can be translated to Horn clauses in a standard way. Nonetheless, the mapping is to some extent syntactic as OWL DL reasoners adopt monotonicity, and adopt the open world assumption, whereas logic programming engines use closed world reasoning and allow non-monotonicity.

The RL profile is a syntactic fragment of OWL 2 DL, but differs from the QL and EL profiles in that its semantics is partially given by a set of rules that extend the RDFS interpretation of valid RDF graphs of this profile. All entailments of OWL 2 DL reasoner over this fragment will be trivially sound and complete, but an OWL Full (or RDFS) reasoner will additionally have to implement the rules to ensure soundness. The OWL Full/RDFS interpretation is an extension of OWL 2 RL that was originally developed by Oracle[45] which, similar to QL, enables efficient OWL reasoning on top of a relational database. A forward-chaining reasoner for this profile has been implemented as part of Oracle 11g. The primary rationale behind the development of OWL 2 RL Full is that existing DL reasoners (Pellet, KAON2) cannot handle reasoning on the often quite substantial amount of entries in databases.

## 3.6 Discussion

The development of the Semantic Web has lead to a number of technologies that, in conjunction, constitute a balanced, layered approach for representing knowledge on the web (cf. Figure 3.1). Despite some initial problems due to its unclear and non-standard semantics, RDFS has proved to be a solid founda-

[44] As defined in RuleML, see `http://www.ruleml.org`, and now RIF, the Rule Interchange Format, `http://www.w3.org/2005/rules/wiki/RIF_Working_Group`

[45] Also known as OWL Prime, or RDFS 3.0, see `http://www.oracle.com/technology/tech/semantic_technologies/pdf/semantic11g_dataint_twp.pdf`.

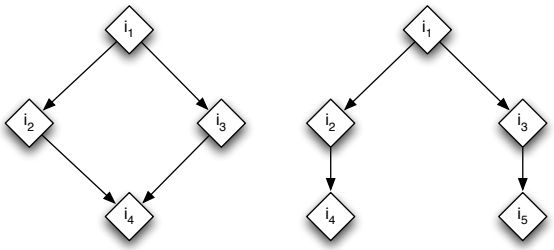


Figure 3.6: Diamond vs. tree model

tion for the now standard representation language OWL. The representation of knowledge on the web posed several requirements for this language in addition to those outlined in Chapter 2, with respect to both semantics and syntax. On the one hand, compatibility of its syntax and semantics with the lower layers in the cake, RDF and RDFS, had to be ensured. On the other hand, it should fit the tradition in knowledge representation, and description logics in particular. Inference on the axioms in ontologies expressed in OWL is *monotonic* to remain unaffected by the addition of new information. OWL therefore adopts the open world assumption. Lastly, efficient algorithms for the language have been shown to exist, i.e. it is *decidable*, and have been implemented in several reasoners.

The above aspects of the web ontology language are a consequence of the way it deals with several trade-offs (Horrocks et al., 2003). The most important of these is perhaps the one between *decidability* and *expressiveness*. In this respect, the development of OWL has been, and still is a cautious one: new features are only added to the language provided that some suitably efficient algorithm is known. Decidability of OWL, as a fragment of first order logic, was therefore only possible by sacrificing expressiveness.

The most prominent hiatus in expressiveness lies exactly where rule based formalisms find their strength. As most decidable description logics, $\mathcal{SROIQ}$ has the tree-model property, i.e. every concept in the logic has a model only if it has a *tree*-shaped model. In a tree-shaped model the interpretation of properties defines a tree shaped directed graph. In its simplest form, the problem is that class descriptions cannot be used to distinguish between the two patterns of individuals in Figure 3.6, i.e. a class description cannot enforce the existence of an owl:sameAs relation between individuals $i_4$ and $i_5$. In short this also means that defining the pattern of Figure 7.4 is theoretically impossible in this language.

Although several decidable fragments of logic programming languages such as Datalog exist that *can* express diamond-shaped models, these are extensions of first order logic which operate under the closed world assumption and thus do not meet the requirement of monotonicity. Furthermore, the development of web-based rule languages has proven to be slow going because of the enormous range of different rule-based languages on the market today. After its start in 2005, the Rule Interchange Format (RIF) working group has only released its first public working drafts in 2008 (including a specification

of its interoperability with OWL).[46]

In general it can be said that any knowledge representation language intended to be used for reasoning necessarily must find a balance between expressiveness and computational complexity: the trade-off is inevitable. In Chapter 7 I discuss the theoretical considerations and practical limitations of this trade-off in more detail. However, whether what *can* be expressed is sufficient or not depends on what *should* be represented.

Despite the occasional drop of the 'O' word, inevitable in discussing a web *ontology* language, this chapter steered clear of any of the pitfalls surrounding the term 'ontology'. Although OWL was presented as a web-based knowledge representation language, the fact that it is called an ontology language is not immediately obvious. The next chapters approach this confusion head-on. Chapter 4 elucidates the different interpretations of the term (and there are a couple), leading to the idea of an ontology as knowledge representation artefact that can be *built*. Nonetheless, an ontology remains a special breed, which becomes evident in the discussion of the issues surrounding ontology construction in Chapter 5.

[46] See `http://www.w3.org/2005/rules/wiki/RIF_Working_Group`.

# Chapter 4

# Ontologies

> "We now begin the science of the properties of all things in general, which is called ontology. (...) One easily comprehends that it will contain nothing but all basic concepts and basic propositions of our a priory cognition in general: for if it is to consider the properties of all things, then is has as an object nothing but a thing in general, i.e. every object of thought, thus no determinate object."
>
> *M. Immanuel Kant (1782–1783)*

## 4.1 Introduction

In Chapter 2, the term 'ontology' was introduced as a moniker for the domain theory of an expert system (Davis et al., 1993, and Section 2.3.2). The functional approach of Levesque (1984) brought us to consider description logics languages as ideal candidates for the representation of these domain theories (Section 2.5.1), and Chapter 3 described a particular member of this language family, the Web Ontology Language, for representing knowledge on the Semantic Web.

For the sake of simplicity, we assumed that the notions of domain theory and ontology stand in direct correspondence. However, this is not the case and despite its success, the term 'ontology' has remained a rather ungainly characterisation of the things it is used to denote. A large number of academic publications revolve around a particular ontology, some ontology language, a methodology for ontology building or a discussion of different *kinds* of ontologies. An invariably significant portion of these papers include some definition of what (an) ontology *is*. Most cited in this context is the definition of Gruber (1993, 1994):

"An ontology is an explicit specification of a conceptualisation."
Gruber (1994)

The apparent convergence between different fields in AI on what an ontology *is*, does not reach very far beyond the superficiality of Gruber's definition. Arguably, taken on its own, a definition is not very helpful. In fact, there are uncountable alternative definitions of ontology, that are equally uninformative when taken out of context (cf. Section 4.4). The problem is, definitions are rather imprecise – e.g. what then is a conceptualisation? – and hide the rationale and perspective that underpin the definition. Perhaps for this reason, the definition of ontology as proposed by one researcher is often subject to heavy criticism from others who have a different background. Definitions can be widely divergent, and ontologies can range from lightweight textual descriptions of some terms to highly formal specifications of philosophical primitives.

It is not wholly inconceivable that the longstanding and still prevalent custom of including Gruber's definition in scholarly articles is a serious indication that we *still don't really know what an ontology is*. Perhaps we don't *want* to know, or at least keep up the appearance that we know what we are doing. As AI is very much an interdisciplinary field, this rather careless attitude has a detrimental effect on the overall quality of 'ontologies' produced in the field – at least when seen from the knowledge representation perspective of the preceding chapters.

## 4.2 Ontologies as Artefacts

McCarthy (1980) first borrowed the term 'ontology' from philosophy to refer to the things that exist in a description of (all) commonsense knowledge. The perspective of philosophy fit well with McCarthy's position that knowledge in an intelligent agent should be based on a small number of principles (see Section 2.2.1). Nonetheless, the term remained only spuriously used in AI until it was adopted by the knowledge acquisition community. And this time, it had quite a different ring to it. No longer it was used to refer to *the* theory of existence, but rather as reflection of the building blocks of a domain theory: *concepts*. Ontologies soon grew into knowledge representation artefacts in their own right.[1]

As we have seen in Chapter 2, separating problem solving knowledge from domain knowledge in knowledge based systems has proven to be a fruitful means to circumvent the interaction problem of Bylander and Chandrasekaran (1987) and improve reusability of knowledge components. Originally, this separation was not intended to exist physically inside a knowledge based system, but rather, the two types should be *modelled* separately. Because a domain model constrains that which exists for a knowledge based system and what it can reason over, it can be said to capture an ontology (Davis et al., 1993). In this

[1] In the following I will use the term *Ontology*, with a capital 'O', to denote the philosophical discipline, and *ontology* to refer to a (formal) construct reflecting some ontological commitments. The word 'ontological' in that sense means 'pertaining to existence'; an ontological *commitment* is a commitment to the existence of something, ontological *status* is some degree of certainty by which an entity is thought to exist.

view, reflected by the definitions of Gruber (1994) and Schreiber et al. (1995), an ontology is part of the *specification* of a knowledge based system:

> An *ontology* is an explicit, partial specification of a conceptualisation that is expressible as a meta-level viewpoint on a set of possible domain theories for the purpose of modular design, redesign and reuse of knowledge-intensive system components.
>
> Schreiber et al. (1995)

Initially ontologies were merely a novel term for the specification of domain knowledge in the form of documentation, schematic diagrams and textual descriptions akin to specifications in software engineering. It is this type of specification that Gruber meant. To emphasise this perspective, he referred to the specification of ontologies as *ontology engineering*.

Schreiber et al. and Gruber consider the ontology as a necessary step in the design of a system; it can be said to be *implemented* in a system. In exactly the same way that problem solving methods are abstract reusable descriptions of *reasoning*, an ontology enables reuse by providing an abstract description of some *domain*. Ontologies can be consulted when selecting a 'knowledge component' that implements some required reasoning services, or when developing a *new* system or component that re-implements that body of knowledge.

Furthermore, an ontology can help guide knowledge acquisition for a domain by providing a conceptual 'coat rack' to which new knowledge can be added. To give an example, a general expert system for medical diagnosis needs to implement (at a minimum) both the standard diagnosis PSM of Figure 2.9 and an ontology of the human physiology. A more specialised expert system could implement a more specific diagnosis PSM, e.g. a causal-dependency based approach (Bredeweg, 1994), or implement a liver disease ontology that extends the physiology ontology. Both the ontologies and the PSMs are not part of the system itself, but belong to its specification. A knowledge component that implements an ontology can be said to commit to that ontology. And different components that commit to the same ontology are more compatible than those committing to distinct ontologies.

The specification perspective on knowledge gradually grew in importance and it was soon recognised that the ontology – as rather abstract specification which is not part of the system itself – can also serve as a means to *communicate* the expertise of not just components, but of a system as a whole. It enables knowledge sharing across both systems and between systems and people (Neches et al., 1991; Uschold, 1996).

In fact, the techniques of knowledge acquisition were increasingly applied to share knowledge between different groups of people: as techniques for *knowledge management* in organisations (van Heijst et al., 1997; Schreiber et al., 2000). The notions of task decomposition and problem solving methods were very useful in the elicitation of organisational goals and business processes. Domain theories could capture the individual expert knowledge of employees, and thereby chart the distribution of expertise over a workforce. This overview was used to signal lacunae, mismatches and overlap in expertise of both persons and organisational units. The elicitation and *alignment* of domain theories

as a *shared vocabulary* was deemed an especially powerful tool for improving the cohesion and co-operation within and between organisations.

Gruber (1993) takes very much the position that ontologies are essentially about *sharing* conceptualisations:

> *Ontologies* are agreements about shared conceptualisations. Shared conceptualisations include conceptual frameworks for modelling domain knowledge; content-specific protocols for communication among inter-operating agents; and agreements about the representation of particular domain theories.
>
> Gruber (1993)

Ontology is thus abstracted away from its initial purpose in knowledge acquisition, and is more about defining some commonly agreed upon vocabulary of terms for the purposes of *standardisation*. Rather than what one might initially expect, this more abstract view on ontologies puts a lot more weight on their development process and methodology. Anyone who has ever been involved in a standardisation committee will know that the interplay between different parties, with different interests can make the development of the standard a cumbersome process. It requires one to be a lot more explicit as to what the intended (or even allowed) use of an ontology is. A non-standard use of some concept definition may influence its interpretation in such a way that it no longer complies with the intended meaning of a term. If an ontology really is about the sharing of a standardised vocabulary, some rules need to be set out that prevent misuse:

> An *ontology* defines the basic terms and relations comprising the vocabulary of a topic area as well as the rules for combining terms and relations to define extensions to the vocabulary.
>
> Neches et al. (1991)

The emphasis on sharing and standardisation sparked interest in three interdependent directions: to ensure quality through a methodological foothold, to enable the physical sharing of ontologies and to facilitate (formal) specification (Neches et al., 1991). The growing importance of ontologies introduced a need for quality assurance; to safeguard extensibility and shareability by formulating a methodology that enforces general design principles and ensures a repeatable development process (see Chapter 5).

Commitment to a shared vocabulary is only possible if the vocabulary itself can be physically shared as well: ontologies should be *portable*. Portability is ensured by using a common specification language. Such a language provides a *syntactic entity* that can be manipulated, copied and referred to. The ONTOLINGUA system of (Gruber, 1993; Farquhar et al., 1997) is an online editor and library of ontologies, similar to the library of problem solving methods of e.g. Breuker (1994, 1997), that supports the storage of "ontologies that are portable over representation systems" (Gruber, 1993, p.1). Although there is no prescribed level of formality of vocabulary specifications (Uschold, 1996, p.6), direct ontology sharing between systems requires at least a structured language with clear semantics. In particular where *knowledge modelling* ontologies are concerned (van Heijst et al., 1997).

It was clear that a structured specification language for ontologies would increase their portability. However, the *purpose* of the specification turned out to play an important role in determining what a useful specification is.

#### Knowledge Representation Ontologies

The knowledge representation community took the specification to mean a *formal* specification that could be used to constrain valid implementations of the ontology. A formal language can be used to ensure *internal consistency* of vocabulary definitions in the ontology; it may sanction *proper extensions* – incorrect extensions of the ontology are inconsistent; and opens the door to automatic *compliance checking* of implemented knowledge based systems – does the system adequately implement the ontology?

Ontologies in ONTOLINGUA were represented using KIF (Genesereth and Fikes, 1992), a formal, highly expressive language for the explicit representation of concepts and relations. It is purely meant as an *inter lingua* and does not itself support standard inferencing. ONTOLINGUA could directly translate from and to several other representation languages, and supported interaction with dedicated knowledge editors such as Protégé using the Open Knowledge Base Connectivity language (OKBC, Chaudhri et al. (1998)). OKBC was an API[2] akin to the contemporary DIG specification[3] for exchanging knowledge bases between a repository and an editor, rather than a knowledge representation language in its own right. Contrary to KIF, OKBC is a language with relatively poor expressiveness.

A language such as KIF can only be used as interchange between relatively compatible knowledge representation formalisms. If some construct from the source language is not available in the target language, or when there is divergence with respect to semantics, translation cannot occur unattended (See Section 5.4). These problems were partially alleviated by the built in *Frame Ontology* of ONTOLINGUA, a representation ontology specified in KIF that defined knowledge representation constructs commonly found in frame-based and object oriented systems. Translation between different languages was only supported through the Frame Ontology. This adoption of the frame language-style of knowledge representation for representing ontologies was perhaps an obvious step, but an influential one at that. Although Gruber presents the Frame Ontology as a mere convenience for ontology engineers over standard predicate calculus, it soon became the default paradigm for specifying AI ontologies.

The Knowledge Representation System Specification of Patel-Schneider and Swartout (1993, KRSS) took this one step further by enforcing a commitment to the frame paradigm. KRSS was developed to establish a common ground between frame-based KL-ONE like knowledge representation languages such as CLASSIC (Brachman et al., 1991) and LOOM (MacGregor and Bates, 1987). Instead of an inter lingua, KRSS is intended as *standard language* for the direct exchange of knowledge representations between such systems. Like KIF KRSS has its own Lisp-style syntax, and was based on the description logics language of e.g. Baader et al. (1991), developed to extend KL-One's formal basis for terminological knowledge representation languages (Baader et al.,

[2] API: Application Programming Interface

[3] DIG: DL Implementation Group, see `http://dl.kr.org/`.

2003, and Section 2.5.1). In this view, the DL language family was thus not just meant for terminological knowledge representation, but for *ontology representation* as well.

Ontologies specified in DL are knowledge representations and can be directly used as knowledge components. At first sight this seems to conflict with the idea that an ontology should be part of the knowledge level specification of a knowledge based system (c.f the quote of Schreiber et al. (1995) on page 67). However, the notion of an ontology as knowledge base does not necessarily imply that it should be incorporated in a knowledge based system *as is*.[4] The knowledge acquisition community was well aware of the developments with respect to description logics, and these languages were certainly not shunned by the more formal minded.

#### Knowledge Management Ontologies

The knowledge acquisition and knowledge management communities, on the other hand, emphasised a software engineering perspective and adopted the schematic diagrams of object-oriented modelling and design, and later the industry standard *Unified Modelling Language* (UML), to express and specify ontologies. In this view, ontologies are primarily meant for human consumption as part of the design of large scale systems. The *Common*KADS methodology used UML-like diagrams extensively for describing task decompositions, problem solving methods and ontologies alike (Schreiber et al., 2000). This approach has been very successful, as for the first time expertise within organisations could be charted and organised in an intuitive manner. The influence of knowledge management during the nineties has certainly contributed to the increasing popularity of ontologies to describe the domain knowledge of experts.

The Conceptual Modelling Language of Schreiber et al. (1994, CML) and $(ML)^2$ (van Harmelen and Balder, 1992) were proposals for structured languages that could be used for the specification of *Common*KADS models. Contrary to $(ML)^2$, CML did not have a formal semantics, but only provided a structured textual notation and a diagrammatic notation for concepts. However, as knowledge management does not require a full specification of an ontology and ontologies could well be just lists of agreed upon keywords or hierarchies of terms, these languages were only spuriously used.

An important application area for knowledge management is to help organisations deal with the enormous amount of information stored across computer systems. At the end of the nineties, ontologies started to become used to physically *index* the information within organisations. Employees were equipped with *user profiles* that expressed their area of expertise in terms of an ontology. Relevant information that matches the profile could then be brought to the attention of the employee. Because indexing documents by hand is an arduous task, data mining and natural language processing technologies were applied to perform automatic *ontology extraction* and *ontology learning*.

[4] In fact, there are several reasons why this is can be technically problematic, cf. Section 7.2

### Ontology Meets the Web

Ontologies were increasingly specified using specialised ontology development tools. Knowledge acquisition tools such as Protégé (Puerta et al., 1992) were adapted for the specification and documentation of taxonomies. As a result, the language in which an ontology was expressed depended more and more on tools. Also, automatically extracted knowledge management ontologies were stored in relatively closed legacy database systems. This turned out to be a significant impediment to reuse, especially considering the growing need for information exchange between distributed information sources over the web.

Most existing initiatives to develop an interchange language for ontologies preceded the development of the web. They were based on the (then prevailing) conception of the web as a relatively slow, but huge local area network, and not the efficient, social, and uncontrollable means for human computer interaction it is today. The growing interest in ontologies sparked a renewed interest in interchange languages, in particular given the possibilities of a new, versatile syntax offered by XML. The SHOE language (Heflin et al., 1999) can be regarded in this light: a simple, frame-based syntactic interchange language for ontologies. Similar lightweight approaches are RDFS and the current SKOS. The DAML-ONT and OIL languages, on the other hand, were more directly influenced by the knowledge representation perspective on ontologies.

The DAML+OIL member submission to the W3C[5] in 2001 was in many ways a package deal that could not be scorned. It offered a full-blown knowledge representation language in the guise of a web-based ontology exchange language. Berners-Lee (1999)'s ideal of a Semantic Web was brought a significant step closer, and once OWL became a W3C recommendation in 2004 it became the de facto representation language for ontologies. However, for those primarily interested in the knowledge *management* aspect of ontologies, the resulting OWL language was somewhat like a Trojan horse: a relatively heavyweight formal language sneaked in via the back door.

Nonetheless, the more informal use of ontologies persists until today, such as in the widespread use of *folksonomies*, and – quite detached from the web – as standard vocabularies for governments and communities of practice. Many of these lightweight knowledge management ontologies are represented using the relatively inexpressive RDFS or SKOS, but a surprisingly large number are in OWL Full as well (though often by accident, Wang and Parsia (2007)).

Since McCarthy (1980) and Davis et al. (1993) borrowed the term 'ontology' from philosophy, the interpretation of the term in AI has shifted from an essential part of the specification of knowledge based systems, to standard vocabularies and full-blown terminological knowledge bases on the web, or even – as we did in Chapter 3 – any OWL file. Nonetheless, not all has been said, as philosophy certainly did not stand by idly while a centuries-old tradition was hijacked by a bunch of computer enthusiasts. The next section describes Ontology as conceived of in philosophy, and Section 4.4 discusses the main differences between the the two views.

[5] See `http://www.w3.org/Submission/2001/12/`

## 4.3 Ontology in Philosophy

In philosophy, the term Ontology is used in the context of the analysis of the fundamental building blocks of reality, the metaphysical study of existence by first principles: what makes that some thing exists, and how can we ascertain the existence of some thing? As Leibniz put it:

> "Ontology or the science of something and of nothing, of being and not-being, of the thing and the mode of the thing, of substance and accident."
>
> Gottfried W. Leibniz, in (Leibniz, 1903, p.512)

Ontology thus concerns the top-down deconstruction of reality as we perceive it: eliminate its accidental appearance and reduce it to its very bare bones. If we look at Kant's description of the 'science of ontology', we can conclude that the method adopted in philosophical ontology is to focus primarily on those things objects in the world have *in common*:

> "...ontology, the science, namely, which is concerned with the more general properties of all things."
>
> Immanuel Kant, in Kant (1997)

It is the commonalities (and disparities) that are the subject of ontological study, and which are used to construct a comprehensive representation of reality. Important also is that it is the study of *general* properties that *all* things have in common, and not of ad-hoc categories. It identifies elements in general which can be applied to account for differences in *particular*. Ontology operates on a meta level with respect to the things in the domain of discourse. For example, instead of studying the properties that make physical entities differ from mental entities, ontology studies what properties are by themselves. This in line with Aristotle's description of Ontology, which, in his sense, tries to answer the question "What is being?", or as Guarino (1997) rephrases it, "What are the features common to all beings?":

> Ontology is the science of being as such: unlike the other sciences, each of which investigates a class of beings and their determinations. Ontology regards "all the species of being *qua* being and the attributes which belong to it *qua* being".
>
> Aristotle, Metaphysics, IV, 1, from Guarino (1997)

Instead of specifying *a* vocabulary, Ontology thus tries to pinpoint *the* vocabulary used to describe things in the world; it usually adopts realism, i.e. the belief that reality exists independently of human observers. It assumes (or even requires) a direct correspondence between the elements in the ontology and entities 'out there'; and is focused at the *primitives* of being. Consequently, an ontology is to capture *directly* the domain of discourse.[6] The high level of abstraction enables a philosopher to reason a priori with respect to the elements

[6] Usually life, the universe and everything

of some ontological theory. These elements, namely, are considered primitives of human thought and reason.

Of course, the results of this study of existence needs to precipitate in some way; it is unavoidable that ontological research results in an entity embodying some 'ontology'. Formal Ontology is a branch of philosophy that addresses this issue by extending Ontology in two ways (Guarino, 1997), i.e. to:

- Capture ontological theory in a formal language, i.e. first order logic, and
- Study the *forms* and modes of being

Seen in this light, it is understandable that to McCarthy using the term 'ontology' was quite natural. His goal of an ontology as formal representation based on a fixed set of basic principles appears to almost seamlessly correspond to the meaning ascribed to the term in formal Ontology. But the knife cuts both ways: the commitment to a formal specification of an 'ontology' submits formal Ontology to the same restrictions as knowledge representation in AI: the *syntactic* form of an ontology influences the quality of an ontology as a *semantic* entity (Guarino and Giaretta, 1995).

### 4.3.1 Problems in Formal Ontology: Semantics and Syntax

The acknowledgement that an ontology can never be untainted by formalism and design choices is perhaps the single most prominent difference between the approaches in philosophy and AI. As discussed in Section 2.4.1, the separation of knowledge types was introduced in the first place to remediate known hurdles such as the interaction problem and the knowledge acquisition bottleneck. Although an ontology was conceived as a knowledge level specification of the domain theory of a knowledge-based system (Davis et al., 1993), it was well understood that even this trick would not shield the ontology as such from its context in knowledge based systems.

For a long time, this dependency between representation and language was deemed of no relevance for philosophy, as Ontology was expressed in the tradition of e.g. Leibniz using the "universal language of rational thought": logic.[7] Nonetheless, there even were philosophical arguments against a purely logical approach. According to Smith (1978), for a formal ontologist, even the use of first order logic to precisely and accurately define philosophical convictions poses a threat. The trade-off Smith sketches is between *overshooting*, and possibly allowing entities that have limited ontological status, and possibly *missing out* on important entities, which would diminish the ontological adequacy of a theory as a whole.

The former solution is of course regarded unacceptable by puritan realists. Namely, an ontology that commits to the existence of entities that do not exist is fundamentally flawed. On the other hand, Smith argues that early formal philosophy was caught in a 'perversion' of Occham's razor. His maxim to not multiply entities without *necessity*, was misapplied as a much stricter practice: not to add entities wherever *possible*. It is furthermore fuelled by a combination of reductionism and pragmatism as in e.g. Frege's work where philosophical

[7] Note that although the current language of choice is first order logic, Leibniz' view was primarily computational.

progress is measured "by the degree to which one can 'explain away' apparent philosophical givens in terms of less controversial entities". According to Smith, this perversion leads to a simplification of the world – the subject matter of Ontology. Not as an inevitable by-product of the use of a formal language, but rather due to the application of an overly simple *mathematical* formalisation. For, what evidence is there to expect that the logical constructs devised by Frege to explain and define mathematical theory are equally well suited to capture ontological theory?

Smith illustrates this practice by positing a school of thought by the name of '*fantology*', the idea that ontological form corresponds to one and two-placed predicates of the form F$a$ and R$ab$ (Smith, 2005). This view confounds the first role of formal Ontology – to capture ontological theory in a formal language – with the study of 'forms of being', by equating ontological form to logical form. Arguably this is problematic, as this conflation allows the application of common operators of logic such as the Boolean and and or, to ontological categories: ontological truth becomes equivalent to logical truth.

In this conception, the predicate F carries the meaning, whereas the subject $a$ is a 'mere meaningless name, a matter of pure denotation' (Smith, 2005). Although nothing in logic prevents us to ascribe meaning to the subject of a predicate, the prevailing philosophical interpretation is that they refer only to individual objects. Furthermore, the predicates themselves are not ontologically neutral (Smith, 2004). For example, the relations is_narrower_than and part_of are certainly not of the same type. Where the former expresses a relation between *meanings*, the latter expresses a structural tie between *universals*.

Smith argues for a system where not the predicates, but indeed the subjects of those predicates carry meaning. The predicates themselves 'do not represent', but rather are what link together variable and constant terms. In this proposal Smith eliminates unary predicates altogether, and restricts the number of allowed relational predicates to a fixed set, containing amongst others subsumption, parthood and dependency relations. A restriction that was also advocated by Breuker and Wielinga (1987). Recall that in the 1970's semantic networks were criticised because their structure was too generic and semantically unclear (cf. Section 2.2.3). The solution was to develop languages that contained a fixed set of knowledge structuring primitives. Though given by different reasons, the proposal by Smith is in fact analogous to this solution.

Guarino (1994) takes a different approach, and proposes to limit the scope of predicates by formulating semantic constraints. These can be used to express the difference between e.g. sortals and non sortals, i.e. predicates that are substantial, e.g. whether some entity is an apple, apple$(x)$, and those that express a mere characterisation, such as red$(x)$.[8] This way, it is thought, a rigourous ontological foundation of the primitives in some knowledge representation language can guarantee a consistent interpretation of theories across different domains.

However, from a knowledge representation perspective, it is unclear how such a priori distinction between predicates on entities is possible. Or at least, how it is different from any other restriction posed in a knowledge base itself – and not in an ontological layer incorporated in the representation language.

[8] Guarino (1994) also introduces *rigidity*. See the discussion on the OntoClean methodology in Section 5.5.1 for a more in depth discussion of this notion.

Whether some predication of an entity meets its formal ontological requirement can only be checked against the *actual* entity *in the domain*, which is not formally specified other than by means of the predication *itself*. For instance, there is nothing 'in' the predicate red that precludes its interpretation as substantial: does $\mathsf{red}(x)$ state that $x$ has the property of being red, or that $x$ *is* the colour red?

In summary, the specification language used in formal Ontology turns out not to be ontologically neutral, but rather has to be used with care. This can be achieved either by having its predicates quantify over parts of the ontology, instead of individuals, or by distinguishing different ontological categories of predicates.

## 4.4 Two Kinds of Ontologies

As said in Section 4.2, use of the term 'ontology' by the knowledge acquisition and representation community did not go unnoticed in formal Ontology. The wider adoption of the term, until then private to a small community, sparked concern as to the place of Ontology in knowledge representation. While in knowledge acquisition, ontology construction was an important but preliminary step in knowledge based systems development, and knowledge management even posed the ability of ontology *extraction*, formal ontologists naturally saw a more prominent role.

Guarino (1994) made efforts to integrate formal Ontology with the notion of knowledge representation languages. He argues for an *ontological level* on top of Brachman's epistemological level. Where the epistemological level provides structure, the ontological level is to constrain the meaning of primitives in a knowledge representation language. Guarino criticises the neutrality of knowledge representation formalisms as regards their ontological commitment. In his view, structured representation languages such as KL-ONE cannot be "distinguished from their 'flat' first-order equivalents" without making the ontological commitments underlying their structure explicit. It should be made clear what it 'means' to interpret binary predicates as roles, and unary predicates as concepts:[9]

> At the ontological level, knowledge primitives satisfy formal meaning postulates, which restrict the interpretation of a logical theory on the basis of formal ontology, intended as a theory of *a priori distinctions.*
>
> (Guarino, 1994, p.444)

Serious efforts to reconcile ontology in AI with its older and wiser namesake were of course welcomed by the knowledge acquisition community, but quite often with a sense of bemusement. For the theoretical considerations brought to bear by the likes of Smith and Guarino seem to be of no direct *practical* relevance in the development of knowledge-based systems, let alone knowledge management. And furthermore, the naive notion of (formal) ontology as a direct reflection of reality was somewhat smirked at by a field that

[9] Ontologies that specify the commitment of a formal representation language are usually called *representation* ontologies (van Heijst et al., 1997).

had been struggling with that very same relation between representation and reality for over twenty years. On the other hand, the use of "ontology", as merely a convenient moniker of some part of a specification in knowledge acquisition methodologies was perceived as rather careless by formal ontologists for which the ontology itself is the primary goal.

The relative positions can be summarised as follows:

- Philosophy's main criticism concerned the lack of theoretical philosophical rigour underlying the *content* of ontologies in AI: domain theories are very often philosophically naive.
- AI, on the other hand, (silently) criticised philosophy's disregard of the fundamental problems in knowledge acquisition and representation, such as the interplay between language and representation, and the interaction problem (Bylander and Chandrasekaran, 1987). AI ontologies are meant to be used for *reasoning* in the context of very mundane problem solving tasks where overly theoretical conceptions are more likely to be a burden than a help.

The apparent incompatibility between principled philosophical and theoretically 'loose' AI conceptions of ontology has in fact quite often led to heated debates about a *proper* definition: a definition that would be compatible with both perspectives and one that could reconcile the positions. There have been several attempts, primarily by Guarino and Giaretta (1995); Guarino (1998) to come to such uniform definition.

One source of confusion has been that originally, both interpretations were vague as to whether an ontology is the specification *itself* or *that which is specified*. In philosophy this was characterised by disregard of the formalism, and in AI by imprecise usage of the term itself. This initial indecisiveness was settled by the well known definition of Gruber (1993, 1994) (see Section 4.1), which distinguishes the ontology, as specification, from that which it specifies, the 'conceptualisation' (Genesereth and Nilsson, 1987). Surely we can conceptualise, or understand, the world in many different ways. An ontology captures a commitment to those parts of a conceptualisation that we deem to exist in reality. This conceptualisation is a priori inaccessible: it only exists "in someone's head" (Uschold, 1996). The explicit specification of a conceptualisation is therefore just as subject to the knowledge acquisition bottleneck as other forms of knowledge representation, and it was consequently acknowledged that a conceptualisation can only be approximated:

> An *ontology* is an explicit account or representation of some part of a conceptualisation.
>
> Uschold (1996), adapted from Guarino and Giaretta (1995)

Taken in this light, Ontology, as the philosophical discipline, endeavours to approximate the a priori conceptualisation that underlies the structure of reality. However, in the context of knowledge-based systems, the notion of ontology is clearly 'disconnected' from reality: the conceptualisation being specified is that shared by one or more experts. No claim is made as to the real

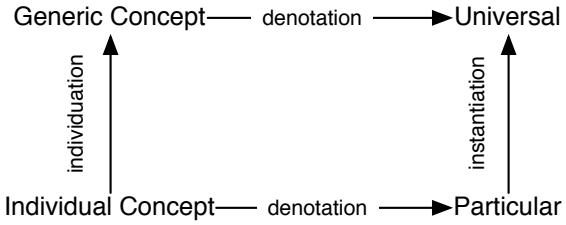


Figure 4.1: Ontological relations in Realism

existence of elements in the ontology. The ontology specifies that which a system 'knows' about:

> An (AI-) *ontology* is a theory of what entities can exist in the mind of a knowledgeable agent.
>
> Wielinga and Schreiber (1993)

In other words, the ontology prescribes what entities can be distinguished, or rather *individuated* inside a knowledge base: an ontology encompasses its *generic concepts*. Recall the relation between a knowledge representation and entities in reality in Brachman's meaning triangle of Figure 2.4.[10] Admittedly, both generic and individual concepts in a knowledge representation can be related to some entity (individual object) in reality through instantiation and denotation, respectively. It is the ontological status and strength of these relations as to which philosophy and AI differ. Firstly, in AI, denotation is a correspondence relation between a concept in a knowledge base and some entity in reality. It is generally true that an individual will only be asserted into a knowledge base given some corresponding entity, but this is not enforced. In fact, individuals are often asserted for purely practical reasons, as mere database keys. Secondly, instantiation of a generic concept by an entity is an even weaker relation in the ontological sense: the entity merely *exemplifies* the generic concept.

These are considerable weaker versions of their philosophical interpretations, especially in comparison to the position of realism. Realism holds the existence of *universals*, properties such as ''*being an apple*' that hold in multiple places, or rather are instantiated by multiple *particulars*. Using KR wording, realism essentially adopts the stance that reality contains properties (entities) denoted by generic concepts (see Figure 4.1). As AI makes no such claims, proposals for definitions of 'ontology' are often accused of adopting the opposite position of *nominalism*, which holds that universals only hold as names.

However, this is a false accusation as has been made clear by more philosophically aware AI researchers. For instance, Genesereth and Nilsson (1987) who coined the word 'conceptualisation' on which Gruber's definition is based,

[10] Keep in mind that Brachman did not distinguish symbol level and knowledge level representation.

explicitly state that no attention has been paid to the question whether the objects in a conceptualisation really exist. They do not adopt realism nor nominalism:

> "conceptualisations are our inventions, and their justification is based solely on their utility. This lack of commitment indicates the essential ontological promiscuity of AI: any conceptualisation of the world is accommodated, and we seek those that are useful for our purposes."
>
> (Genesereth and Nilsson, 1987)[11]

The keyword here is '*utility*'. Philosophical ontologies are not intended to be *used* in the operational sense of the word. They are not intended to be a part of some knowledge based system, but rather reflect a formal commitment to some philosophical theory. It raises the question as to whether the notions of philosophical and AI ontology are compatible: *can* a philosophical ontology be used in practice? This assumption is often implicit in philosophically inspired formal ontologies (Grenon, 2003; Masolo et al., 2003). But, as discussed in Chapter 6 it may not always hold (Hoekstra et al., 2007, 2008).

## 4.5 Discussion

The mixture of philosophical and knowledge representation perspectives involves a trade off between an ontology as *knowledge representation artefact* and as *theory of existence*. According to the first view, the quality of an ontology does not relate to philosophical adequacy, but rather to its suitability to play a role in problem solving (Genesereth and Nilsson, 1987) and knowledge reuse. The design choices inherent in the knowledge based system determine what the ontology contains, and no particular restrictions hold as to its shape or form. The main requirement is that the ontology should adequately capture the system's domain theory and be neutral with respect to its control knowledge (Chapter 2).

The philosophical perspective, on the other hand, poses additional restrictions on the *content* of ontologies. First off, it is clear that the purely philosophical view of Formal Ontology cannot easily be reconciled with the knowledge representation view because where the former sees language primitives as the primitives of existence – extending an ontology equates to extending the language – the latter restricts language primitives based on pragmatic, epistemological and computational considerations (Levesque and Brachman, 1985, and Section 2.5.1). The discrepancy between the two views is evident in e.g. Bera and Wand (2004)'s refutation of OWL as ontology language. As discussed in the preceding chapters, this is the price one has to pay for the practical application of ontologies in reasoning.

The development of ontology representation languages inspired the use of ontologies as a readily available resource for knowledge based reasoning. These *knowledge representation ontologies* are specified using their own carefully crafted representation language (OWL DL), and inference is supported by highly optimised reasoners. The ontological perspective is partly ensured by adopting the DL paradigm as it only sanctions inference that is ontologically relevant: *consistency* checking of axioms in the ontology, *classification* of concepts as

belonging to more generic categories, and *instance* checking of individuals as denoting instances of some concept. This way, the semantics of the language makes the representation of an ontology correspond more directly to an explicit set of ontological commitments: the ontology cannot commit to more than what is inferred by the reasoner (cf, Davis et al. (1993) and Section 2.3.3).

Despite their importance, these are mere preconditions for the development and use of *good quality* ontologies: two OWL axioms do not make an ontology. Calling any domain theory an ontology does not do justice to the claim that underlies the adoption of the term in the first place: the ontology expresses a theory of existence. Of course, Davis et al. are entirely right in stating that a knowledge representation is "a *set of ontological commitments*" (Davis et al., 1993, p.3), but it expresses *other* commitments as well. For instance, Clancey (1983) identified causal rules as part of the domain theory of MYCIN (see Section 2.3.2). Though certainly not part of the application's control knowledge, causal rules reflect an *epistemological* rather than ontological perspective. The distinction between the two is discussed in more detail in Section 5.5.2.

Because the quality of ontologies depends on subtle, and competing requirements, their development is a delicate task that involves a large number of important decisions. Decisions that carry additional weight when considered in the light of knowledge sharing and reuse. Of course, an ontology engineer needs to decide not only which concepts and relations to include, but also the level of detail in which they are defined. Furthermore, every definition should adequately cover the intended meaning of a concept: the traditional knowledge acquisition bottleneck (Feigenbaum, 1980).[12]

This chapter presented an overview of the different conceptions regarding what 'an ontology' is. It distinguishes three views:

- *Knowledge Management Ontologies* are (structured) vocabularies developed for sharing and reuse of information within organisations. Languages suitable for representing these ontologies can be lightweight, as no (expressive) reasoning is required. Examples are RDF/RDFS, SKOS, UML or Topic Maps.

- *Knowledge Representation Ontologies* are reusable terminological knowledge representations that specify the part of a domain theory that directly reflects its ontological commitment. Languages suitable for representing these ontologies incorporate a trade-off between expressiveness and decidability, to support ontology-based reasoning services within knowledge-based applications. Examples are description logics, and most notably OWL DL.[13]

- *Formal Ontologies* are formal specifications of an ontological theory in philosophy. Languages suitable for representing these ontologies are highly expressive and involve a minimal ontological bias as regards their language primitives, such as first-order logic.

[12]The knowledge acquisition bottleneck is often misunderstood as the high threshold in effort before knowledge representation starts to pays off, and practical reasoning problems can be solved. However, in Feigenbaum's original reading it rather refers to the general difficulty of correctly extracting expert knowledge into a knowledge base, see Section 2.4.1 on page 29.

[13]Note that some methodologies, e.g. van Heijst et al. (1997) use the term 'representation ontology' to refer to an ontology that defines the primitives of a knowledge representation language. This is not what is intended here.

Although the ontology types and languages do not correspond directly, confusion may arise when one of the language paradigms is applied in the representation of a different ontology type. The following chapter outlines requirements and methodological guidelines for the construction of ontologies needed to ensure both their reusability and ontological nature. Chapter 6 describes the construction of a core ontology for the legal domain that aims to maximise these factors.

# Chapter 5

# Ontology Engineering

> A common mistake that people make when trying to design something completely foolproof is to underestimate the ingenuity of complete fools.
>
> *Douglas Adams*

## 5.1 Introduction

Over the years several design principles have been cast in ontology engineering methodologies to improve the quality of ontology development. And fortunately, ontologies are not necessarily drafted from scratch. The widespread availability of ontologies on the web has opened the door to the adoption of pre-existing general ontologies. By providing an initial structure and a set of basic concepts and relations, these ontologies can be a valuable jump start for more specific ontology development.

On the other hand, an ontology is not just *any* terminological knowledge base, but rather embodies a specific definitional perspective. Where the representation of terminological knowledge is based on Minsky's notion of a frame – concepts are defined by the typical context in which they occur – an ontology refines this notion and needs to distinguish between the inherent and accidental properties of concepts (Breuker and Hoekstra, 2004c; Guarino and Welty, 2002; Bodenreider et al., 2004). Although a typical property such as 'position' is clearly relevant from a knowledge engineering perspective, ontologically speaking it is usually a side issue: changing the position of an object does not make it intrinsically different. This *epistemological promiscuity* of terminological knowledge representation has led to the formulation of several design principles, such as in ONTOCLEAN (Guarino and Welty, 2002, 2004) and in Breuker and Hoekstra (2004c); Breuker et al. (2004); Hoekstra et al. (2008).

A more recent development is the identification of *design patterns* that are meant to overcome some of the limitations in the reuse of existing ontologies and the application of design principles. Where the former are frequently quite large and heavyweight, and are therefore hard to mould and extend to a usable domain ontology, the latter are quite abstract and difficult to translate into concrete definitions in the ontology. Design patterns offer a middle ground between the two, where the methodological principles are made concrete in manageable 'bite-sized' chunks (Gangemi, 2005, a.o.). Furthermore, these patterns can assist in tackling problems related to the ontology representation language of choice (Hoekstra and Breuker, 2008, a.o.).

This chapter gives an overview of each of these four approaches and discusses their strong points and weaknesses.

## 5.2 Criteria and Methodology

During the mid-nineties, experience in several large scale ontologies led to a widespread consensus that the ontology engineering field was in need of methodological grounding. Independently, the design principles and experience of several ontology efforts were put to paper. Most influential, in this respect are the experiences of the TOVE (Grüninger and Fox, 1995, TOronto Virtual Enterprise) and Enterprise ontologies (Uschold and King, 1995). Both ontologies were developed to enable the specification and sharing of descriptions of business processes. On the basis of experiences in ontology development for the Ontolingua system in Gruber (1993), Gruber identified a trade off between five design criteria (Gruber, 1994):

**Clarity**
An ontology should effectively communicate the intended meaning of defined terms. Definitions should be objective, formal (if possible), and documented with natural language.

**Coherence**
An ontology should sanction inferences that are consistent with the definitions.

**Extensibility**
An ontology should be structured such that it can be extended and specialised monotonically, i.e. without needing to change the ontology itself.

**Minimal encoding bias**
The conceptualisation should be specified at the knowledge level, without depending on a particular symbol level encoding, e.g. constructions for convenience of notation or implementation.

**Minimal ontological commitment**
An ontology should only enforce the minimal commitment necessary to support the intended knowledge sharing activities.

The trade-off lies in the fact that these criteria cannot be considered independently. The clarity and coherence of an ontology benefits from a formal specification. But every formal specification involves an inherent concession

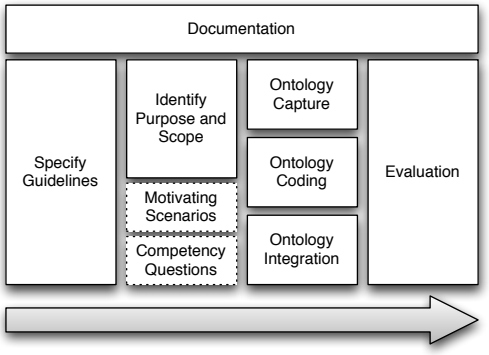


Figure 5.1: Phases in the methodology of Uschold and Grüninger (1996)

with respect to the encoding bias of the language used. With the arrival of the Semantic Web and OWL, most of these parameters have to some extent been fixed (see Chapter 3).

Although use of OWL as ontology representation language involves a significant commitment to a particular symbol level encoding, the encoding itself is designed for maximal clarity and coherence. Extensibility is ensured through its open world assumption and imports mechanism, and since OWL is currently *the* standard for sharing ontologies across the web, it is furthermore likely that a commitment to this language facilitates rather than impedes sharing. Nonetheless these design criteria are not automatically upheld once we adopt the built-in restrictions of OWL: there is no silver bullet. The language leaves a substantial amount of legroom at the conceptual level.

In Uschold (1996); Uschold and Grüninger (1996) experiences in developing the TOVE and Enterprise ontologies were combined in a skeletal methodology for ontology development that incorporates the principles set out by Gruber (1994). Rather than as a precise step-by-step instruction for ontology development, the methodology of Uschold (1996); Uschold and Grüninger (1996) is illustrated in terms of its main characteristics (see Figure 5.1). First of all, they advocate the description of a clear set of guidelines that govern the development process as a whole. These guidelines may range from relatively straightforward agreements, such as naming conventions, to the basic principles of Gruber, and even to formal editorial voting procedures. A clear example of the latter can be found in the OBO Foundry (Barry Smith, 2007); a coordinated effort to develop and integrate multiple ontologies in the biomedical domain.[1] Changes to one of the ontologies in the foundry need to be accepted by an editorial board, and possible overlaps between ontologies are solved via arbitration. One of the points stressed by Uschold and Grüninger is to have guidelines in place for the development of *documentation* during development. It can be quite hard to reconstruct the reasons underlying a particular modelling decision, and having documentation in place at an early stage increases the chance that similar problems are solved in a similar fashion.

[1]The Open Biomedical Ontologies Foundry (OBO Foundry) is freely accessible online, see `http://obofoundry.org`.

In the second phase, purpose and scope of the ontology are identified. Having a clear understanding of why the ontology is developed, its possible use and users, gives a handle on the design decisions necessary in consecutive phases. Grüninger and Fox (1995) advocate the description of motivating scenarios that explain possible use of the ontology and use these to formulate a set of competency questions that should be answerable given the terminology to be defined in the ontology. The purpose of an ontology should not be conflated with the task-dependence of knowledge based system development. In principle an ontology does not sanction a particular use of its contents, but it may be designed to fit a purpose *as a whole*, i.e. how the ontology is used in relation to other knowledge components. The scope of an ontology is its maximal potential coverage in a domain, that is all knowledge inferable from its axioms. Scope therefore only partly relates to the size of the ontology.

Ontology building is the next phase and consists of an interplay between three closely related tasks: capture, coding and integration. Ontology capture is the identification of key concepts and relations, the production of precise definitions in natural language, and the important task of choosing the proper term to refer to these ontology elements. Natural language definitions are in fact the primary source of documentation for people unacquainted with the resulting ontology. Ontology browsers such as Ontolingua and the more recent TONES Repository browser,[2] as well as the mainstream ontology editors Protégé and TopBraid present axioms and local annotations in the same view.

Once the main concepts and relations have been identified and described, they are to be represented in some language. Ontology coding thus naturally involves a commitment to a particular knowledge representation formalism; a commitment that needs to be well motivated because the level of formality has a direct effect on the diligence required of the ontology engineer. The choice for a particular knowledge representation language involves two important commitments (see Chapter 4):

- A *direct* ontological commitment to the structure imposed on the world by the primitives and semantics of that language, and secondly,
- a *more formal* definition of a concept implies a *stronger commitment* to a particular interpretation.

The most suitable level of formality is determined by the purpose of the ontology. A knowledge representation ontology that is to play a part in reasoning requires a more limited formal language than a formal ontology in philosophy, see Section 4.2. Similarly, a knowledge management ontology that is merely used to annotate resources for the purposes of lightweight information retrieval inspires less concern for logical consequence, but still calls for machine interpretability. The developer of an ontology that merely expresses an agreed upon vocabulary to be used within a community of practice might even skip the coding step entirely and suffice with a list of definitions in natural language only. For example, it may not be very useful for languages such as the Dublin Core metadata initiative[3], FOAF[4] and RSS[5] to include exact ontologic-

[2]See `http://owl.cs.manchester.ac.uk/repository/browser`

[3]See `http://dublincore.org/documents/dces/`

[4]Friend Of A Friend, see `http://www.foaf-project.org/`

[5]RDF Site Summary, see `http://en.wikipedia.org/wiki/RSS_(file_format)`. RSS currently has various dialects.

ally concise, rigourous formal definition of the exact meaning of concepts such as 'author', 'friend' or 'news'. On the other hand, a formally specified biomedical ontology may be used in conjunction with a standard reasoning engine to automatically classify proteins into functional classes (cf. Section 7.4).

The third task in the building phase is to take advantage of knowledge reuse by integrating the ontology with already existing ontologies. A natural step, it may seem, but in practice this can be quite demanding. First of all, it requires a thorough understanding of the imported ontology. This is because importing another ontology does put a strain on the principle of minimal ontological commitment: it may significantly extend the size of your ontology. The mere fact that some other ontology contains a proper definition of some concept that is within scope, is often not enough reason to include it. Are the *other* concepts in the imported ontology within scope, and does the imported ontology support our purpose? It should be clear what the assumptions underlying the other ontology are. Secondly, if formally specified, the imported ontology must at least be formally compatible with the knowledge representation formalism selected in the coding task. Arguably, as Uschold and Grüninger (1996) point out, the three tasks should not be performed in sequence. The choice for a particular formalism in part depends on the availability of suitably specified other ontologies.

In the final phase the ontology is evaluated with respect to the requirements specified earlier. If a formal language is used, the competency questions can be reformulated in this language to check compliance and coverage. Secondly, if deviations from the initial outset become clear, it should be checked whether these are well motivated.

### 5.2.1 A Social Activity?

Most methodologies followed in the footsteps of earlier experiences in knowledge acquisition, such as the CommonKADS approach of Schreiber et al. (2000). The methodologies of Uschold and King (1995); Fernández et al. (1997) invariably adopt a typical knowledge acquisition approach aimed at maximising coverage. Expert interviews, brainstorming sessions and text analysis are used to gather as much information about the domain as possible. This information is then used as the resource material for the actual construction of the ontology.

However, inspiration to reach a mature ontology engineering methodology was sought in adjacent engineering fields in computer science as well. The METHONTOLOGY methodology of Fernández et al. (1997); Fernández-López and Gómez-Pérez (2002) incorporated best practices from software engineering methodologies. In particular, Fernández-López and Gómez-Pérez (2002) regard ontology engineering as a software development process, and introduces the notion of an ontology development *lifecycle*: rather than building ontologies from scratch, it emphasises development as a continuous refactoring of already existing ontologies.

When Gruber formulated his criteria, the most prominent use case for ontologies was the ability to share and reuse knowledge across both systems and communities. The prevalent expectation was therefore that ontologies would be *large scale* and built by *many different people* in concert. Since ontologies are intended to capture a shared view on some domain, their development is much

akin to a process of *standardisation*. According to this view, ontology specification is about reaching consensus on the meaning of terms; it is a *social activity*.

Considering some of the most prominent ontologies we see today, this expectation is only partially fulfilled. A first observation is that ontologies vary widely on the scale of the standardisation involved. On the one hand we *can* distinguish ontologies that are the result of large scale standardisation efforts. On the other hand, most ontologies currently available are relatively small, or the result of an academic exercise of only a few people. Secondly, formal ontologies used and developed by larger communities appear to exist only for scientific domains. The obvious examples that come to mind here are the efforts in the life sciences mentioned earlier, such as SNOMED, the Gene Ontology and GALEN. This is not very surprising as (like ontology) science is all about primitives; about discerning entities in reality by sometimes literally splitting them up into smaller bits. Both Mendeleev's periodic table of chemical elements and the Linnaean taxonomy of species are some of the earliest ontologies and were developed to capture, or rather facilitate the expression of a scientific theory.

It is hard to find a large scale ontology that *is* the result of a community effort. General ontologies, such as the Basic Formal Ontology (BFO, Grenon (2003)), the Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE, Masolo et al. (2003)), Sowa's top ontology (Sowa, 2000) and the LKIF Core Ontology of Basic Legal Concepts (Breuker et al., 2007; Hoekstra et al., 2007, this volume) share a formal characterisation but were specified by only a small number of people.

The Standard Upper Ontology (SUO)[6] of the IEEE is perhaps the only example of a community-driven ontology building building effort aimed at a formal characterisation of a broad range of concepts, even though its strong points lie in predominantly physical domains as engineering and physics. Unfortunately, its development stalled when the working group was unable to resolve several hot issues via the mailing list. At the heart of this debate lay the proposal for a Standard Upper Merged Ontology[7] (SUMO) by Niles and Pease (2001), who had gathered several existing proposals (e.g. the top ontology of Sowa) into a single framework. Unfortunately, SUMO was rejected as a consensus ontology, and recast as Suggested Upper Merged Ontology, again developed by only a handful of people.

Surprisingly enough, the same holds for some of the most popular standard vocabularies used across the web today. FOAF, Dublin Core and RSS were initially specified by a small group of people. Although they tend to be limited in scope and have relatively lightweight semantics, their development as standards only happened after they were already adopted by scores of users. Because of this, the specification of e.g. revisions to Dublin Core *were* a community effort: several parties had built-up a vested interest in the language. The result is a collaborative process which is just as slow going and burdensome as was the development of the infinitely more rigourous and extensive SUO.

[6] See `http://suo.ieee.org/`

[7] See `http://www.ontologyportal.org/`

#### On Weak Hearted Surgeons

To recapitulate, the emphasis on standardisation did not have the expected result. True, several ontologies are currently used as standard vocabularies, but these were not developed through a *process* of standardisation, rather they were *adopted* as standards or reflect an already existing standard. It is furthermore unclear whether ontology engineering as social activity has any beneficial effect on the quality of the resulting ontology. Are ontologies based on consensus 'better' ontologies? Experience shows that this is not necessarily the case, if only because different parties may not even agree on the purpose of the ontology. For instance, if the ontology is to "facilitate knowledge exchange between applications" this may be interpreted as "the ontology should cover the knowledge expressed by my application" by individual contributors. The result is that politics get in the way of quality, and definitions are tweaked for the sake of consensus. This is evident in e.g. FOAF which includes a wide range of properties to refer to the name of persons: foaf:name, foaf:givenname, foaf:firstName, foaf:family_name and foaf:surname. But it was also what eventually stopped SUO progress; a consensus ontology was drafted which was deemed unacceptable by some on theoretical grounds.

A second impediment to ontology construction as social activity, is that contrary to standardisation in general, we are not dealing with a level playing field. The formal definition of terms is often quite technical, which means that not every interested party can fully participate in the definition process. Who would like to confront a subject matter experts with the intricacies of the interaction between global domain and range restrictions and existential restrictions on classes? Furthermore, the activity is otherwise constrained as well: a formal representation language typically puts limitations on what *can* be expressed, the most general restriction being consistency. For a long time it was held that ontology engineering required ontologies to be specified at the knowledge level, rather than directly in a representation formalism. Of course, the knowledge management view of ontology is compatible with this perspective, and most ontology engineering methodologies do not really pay much attention to this language bias. However, they do take into account the effects of order dependence in taxonomy construction.

## 5.3 Categories and Structure

Building an ontology starts from a general idea of its purpose and scope. The way in which this overview is translated into a concrete set of concept definitions is not trivial. In particular, the process is order dependent and traditional top-down and bottom-up approaches introduce a bias. Working top-down by starting with a set of general terms, as traditionally practised in metaphysics, risks imprecision or rework: more specific concepts may not directly fit the general ones and provide an almost unlimited source for revisions to the top level. On the other hand, naïve bottom-up specification risks the definition of many detailed concepts that turn out to be unimportant, or incompatible at a later stage.

Hayes (1985) proposes an approach to the development of a large-scale knowledge base of naive physics. He rejects top-down construction in favour

| | | |
|---|---|---|
| *Superordinate* | Animal | Furniture |
| *Basic level* | Dog | Chair |
| *Subordinate* | Retriever | Rocker |

Table 5.1: Examples of basic-level concepts (Lakoff, 1987, p.46).

of the identification of relatively independent *clusters* of closely related concepts. This allows one to break free of the often demanding rigour of taxonomic relations that inevitably forms the backbone of any ontology. By constraining (initial) development to clusters, the various – often competing – requirements for the ontology are easier to manage. Clusters can be integrated at a later stage, or used in varying combinations allowing for greater flexibility than monolithic ontologies. The idea of clustering information was later adopted and refined by Uschold and King (1995), who propose a *middle-out* approach where construction proceeds up and downwards from a relatively small set of *basic* notions. These basic notions are grouped into clusters, and should be defined before moving on to more abstract and more specific terms.

### 5.3.1 A Basic Level?

The notion of this *basic level* follows the theory of categorisation in human cognition of Lakoff (1987). It is based on the idea that certain concepts are not merely understood intellectually; rather, they are *pre-conceptual*, used automatically, unconsciously, and without noticeable effort as part of normal functioning. This *functional embodiment* means that concepts used in this way have a different, and more important psychological status than those that are only thought about consciously. Basic level concepts are functionally and epistemologically primary with respect to (amongst others) knowledge organisation, ease of cognitive processing and ease of linguistic expression. This *basic-level primacy* makes that we generally use categories at this level to compose more complex and specific categories, and can create more general categories by explaining their differences and correspondences. In other words, categories are organised so that the categories that are cognitively basic are 'in the middle' of a taxonomy. Generalisation proceeds 'upwards' from this basic level and specialisation proceeds 'downwards' (see Table 5.1). In fact, the basic level transcends multiple levels of abstraction, e.g. 'bird' is basic where 'mammal' isn't. Research in linguistics, most notably Ogden (1930)'s *Basic English* – the inspiration for Orwell's fictional language *Newspeak* in his novel '1984' – has indicated that this holds for words as well: about 90 percent of the English vocabulary can be meaningfully described using the other 10 percent; some 840 words.

This seems quite a sensible approach: it is easy to see how basic level concepts can be used as cognitive primitives for expressing more general and specific concepts. A basic level concept indicates the most prominent characteristics of a specific concept as in 'A retriever is a dog that is bred to retrieve game for a hunter', and can be used as prototypical examples in the definition of more general concepts: 'Animals are dogs, cats, mice, birds, . . . '.

Lakoff's basic categories give us intuitive building blocks to derive and cluster more complex notions. They carry a lot of weight and form the conceptual foundation of an ontology. However, the primitive, pre-conceptual

**endless loop**, n. See loop, endless.
**loop, endless**, n. See endless loop.

Table 5.2: The difficulty to define basic concepts, from Pinker (2007, p.12).

nature of the basic level means that representing basic categories *themselves* is certainly not as straightforward. Our knowledge of basic concepts such as dog, chair, hammer, father is largely implicit and hard to break down into smaller components, exactly because they are primitive.

Consider the following definitions (taken largely from Wikipedia):

- A *dog* is an animal that has four legs and barks. Animals are dogs, cats, mice, birds.... A leg is something you use to walk with. Barking is the sound dogs make.
- A *chair* is a kind of furniture for sitting, consisting of a back, and sometimes arm rests, commonly for use by one person. Chairs also often have four *legs* to support the *seat* raised above the floor.[8] Seat can refer to chair.[9] A back is something you can lean against. An arm rest is something you can rest your arm on while sitting.
- *Father* is defined as a male parent of an offspring.[10] A parent is a father or mother; one who sires or gives birth to and/or nurtures and raises an offspring. [11]
- A *hammer* is a tool meant to deliver blows to an object. The most common uses are for driving nails, fitting parts, and breaking up objects.[12] [...] a nail is a pin-shaped, sharp object of hard metal, typically steel, used as a fastener. [...] Nails are typically driven into the workpiece by a hammer.[13]

It seems that the more basic a concept is, the harder it is to cast its description in terms of other concepts (cf. the quote from Pinker (2007) in Table 5.2). Often definitions of basic concepts will be self-referential and form a tautological trap from which it is very hard to escape. We have seen in Chapter 3 that the definition of a concept in knowledge representation languages such as OWL can only be expressed as (restrictions on) relations with other concepts. This means that, inevitably, the meaning of a concept is determined by context. The basic level theorem does not take this into account. Although basic concepts are *perceived* as primitive, they too can only be defined in this manner: that is, either in terms of other basic concepts or (inevitably) by reference to more generic or specific concepts.

The main problem of traditional top-down and bottom-up approaches was that the commitment to a set of generic or specific concepts introduces a bias that may lead to an ill fit between the resulting ontology and the intended

[8] From `http://en.wikipedia.org/wiki/Chair`
[9] http://en.wikipedia.org/wiki/Seat
[10] From `http://en.wikipedia.org/wiki/Father`
[11] From `http://en.wikipedia.org/wiki/Mother`
[12] From `http://en.wikipedia.org/wiki/Hammer`
[13] From `http://en.wikipedia.org/wiki/Nail`

representation. But it turns out that this drawback is only partially lifted by the middle-out approach. Basic level concepts introduce a similar bias as they too can only be defined in context with other concepts. This context is largely implicit and inaccessible because of the cognitive primitive nature of the basic level.

In short, basic concepts cannot be defined prior to other concepts as the middle-out approach suggests. It is rather the other way around. Basic level concepts give a 'feel' of relatedness and context. They are an ideal entry point to carve up the domain into relatively independent clusters, and play an important role in the definition of less basic notions. Lakoff's basic concepts should therefore be seen as being at the core of Schank and Abelson's scripts and Minsky's frames (Section 2.2.4). Regarded as independent entities they are atomic. They do not carry any meaning, but rather function as meaning providers to surrounding notions. Their own meaning, however, can only really emerge through their use in other definitions.

### 5.3.2 Beyond Categories

To be frank, this is as far as ontology engineering methodologies go. They do not really venture beyond well-meant, but gratuitous advice such as 'be sure of what you want before you start', 'document your decisions well, you might forget them later' and 'if the problem is too big, chop it up into smaller chunks'. The decomposition into various phases therefore plays the role of the user manual no-one ever reads.

Admittedly, several suggestions *are* genuinely helpful and convey a deeper understanding of what knowledge engineering entails. Ontology engineering is often called an art or craft, rather than a science or proper engineering field such as mechanical engineering or software engineering (Fernández-López and Gómez-Pérez, 2002). Although this claim is often cited, it is never really substantiated. What it does hint at, however, is that somewhere along the line something magical happens. Apparently some inaccessible mental skill only found in an experienced knowledge engineer can turn some bit of knowledge into an adequate representation. The adoption of Lakoff's basic level by Uschold and King was a conscious move to apply insights from cognitive science to lay bare a part of this process.

We have seen that basic level concepts can help in the formation of clusters and the construction of definitions of non basic concepts without the straitjacket of top-down and anarchy of bottom-up development. However, there is no methodological support for the way in which the definitions in an ontology are specified in practice. Ontology construction is to proceed 'upward' and 'downward' from the basic level, but how?

## 5.4 Ontology Integration

Ontologies are rarely constructed without at least a pre-existing conception of that which is to be represented. They can be meant to directly express an existing theory, as we have seen in the biomedical domain, or capture some body of expert knowledge. Even in the least restricted case, where an ontology is meant to represent common knowledge, that which is to be represented is

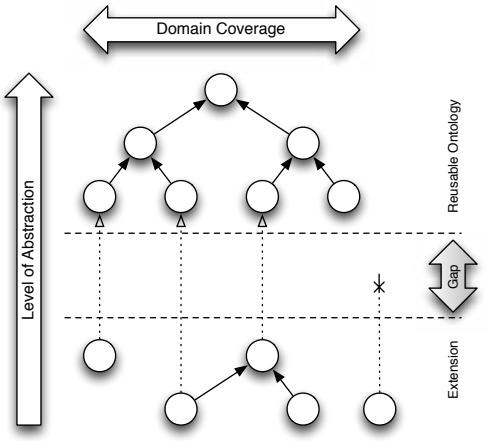


Figure 5.2: Dimensions in ontology reuse

relatively well understood. Nonetheless, ontology development is hard, for how to reliably translate our knowledge into a set of formal axioms?

As discussed, one of the main roles of ontologies is to ease knowledge acquisition by providing readily usable concept definitions. The same bootstrap can be applied to ontology development itself; existing ontologies can be a viable resource for ontology construction in the form of predefined concepts and relations. These definitions convey ontological commitments that capture important ontological choices (Masolo et al., 2003). When these choices are in concert with our requirements, adopting and extending an existing ontology relieves us from the burden of micromanaging these choices ourselves.

One solution is thus to rely on an already existing ontology to provide initial structure: the use of a more general ontology to support knowledge acquisition. This is in fact partly addressed in the ontology integration step of Uschold (1996); Uschold and Grüninger (1996). Reusing and extending an existing ontology solves the structure problem of blank slate ontology development as the existing ontology can be used as a *coat rack* for new concept definitions (Schreiber et al., 1995).

Klein (2001) gives an overview of several issues related to ontology reuse and distinguishes between ontology *integration*, where two or more ontologies are combined into a single new ontology, and *reuse* where the ontologies are kept separate. Both approaches require the ontologies to be *aligned*; the concepts and relations that are semantically close should be related via equivalence and subsumption relations. Since Klein focuses on methods to combine multiple already existing ontologies, this alignment may involve significant modification to the ontologies involved in case they overlap. However, it is not uncommon to modify a source ontology in its reuse as basis for an entirely new ontology. For instance, Pinto and Martins (2000) stress a thorough evaluation of candidates for reuse, and see the modification of those ontologies on the basis of this evaluation as an integral part of ontology development.

Modifications can be necessary if the combination of ontologies results in a heterogeneous system where mismatches may exist at either the *language level*

or the ontological or *semantic* level (Klein, 2001; Visser et al., 1997). Where the former concern incompatibility with respect to how concepts and relations *can* be defined, the latter signals differences in the ontological assumptions about the represented domain. Language level mismatches are not merely syntactic, such as in the incompatibility of the KRSS lisp-style syntax with the RDF/XML syntax of OWL, but may be more fundamentally related to the expressiveness of different languages (Chalupsky, 2000). Since a knowledge representation formalism can be uniquely identified by its position in the traditional trade-off between expressiveness and tractability identified by Levesque and Brachman (1987),[14] the alignment of two representations in different languages invariably involves either choosing one position over the other, or finding a common middle-ground. In the worst case, when translating to a less expressive language this will lead to loss of information. Translating to a more expressive language is less problematic, but will nevertheless require mapping and rewriting of many knowledge representation idioms. However, in many cases the languages overlap; e.g. one language may be more expressive in one area and sport qualified cardinality restrictions, where the other does not do so, but allows for property chain inclusion axioms.

Semantic mismatches are more subtle, and are sometimes even the main reason for starting the alignment. For instance when two ontologies differ in coverage and granularity, their conjunction will increase both.[15] Ontologies may also differ in their ontological stance (Masolo et al., 2003), e.g. choosing a 4D over a 3D perspective on identity persistence through time (Haslanger, 2003), or with respect to more practical modelling paradigm and conventions (Chalupsky, 2000), such as interval versus point-based representation of time, or representing property values as constants or datatype values. Resolving mismatches between ontologies can to some extent be facilitated using techniques for (semi) automatic ontology merging and alignment, such as in the PROMPT system of Noy and Musen (2000).[16]

Although the problem of integrating existing ontologies is more intricate than direct ontology reuse for the development of a new ontology, the possible mismatches give insight in the kinds of things one should look for when choosing a suitable candidate for reuse. This choice thus depends, besides on its overall quality, on four suitability factors: *domain coverage* of the ontology, its *level of abstraction* and *granularity*, its *representation language* and the *level of formality*. Figure 5.2 shows the interaction between these factors.

Domain coverage corresponds to the breadth of the reused ontology. Definitions in an ontology are reused by linking into its structure: axioms in the new ontology are expressed in terms of the existing vocabulary. If the domain coverage of the reused ontology is sufficient, all new definitions can be connected in this a way. Limitations in coverage require either one of two measures: the addition of a new concept at a higher level of abstraction, or the reuse of another ontology that *does* provide the necessary definition. Both solutions are not without problems. First of all, extending the new ontology with more general concepts violates the principle of *stratification*, the idea that different levels of abstraction should be separate, and reuse should accumulate in lay-

[14] See also Section 3.4

[15] This holds especially when constructing an entirely new ontology, as it does not (yet) *have* coverage or granularity.

[16] See Klein (2001) for a comprehensive overview.

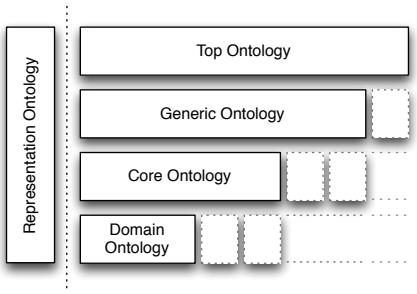


Figure 5.3: Ontology types based on van Heijst et al. (1997)

ers. Adding generic concepts to the new ontology may lead to a necessity to augment already existing concept definitions in the reused ontology. For instance, adding the common abstract notions of 'occurrent' and 'continuant' when reusing an ontology that does not make this distinction requires disjointness axioms between existing concepts. This may be quite sensible from the standpoint of direct reuse, but breaks compatibility with other ontologies that reuse the same ontology. Secondly, extending coverage by reusing another ontology is only feasible in case both reused ontologies are aligned with each other.

A typical way in which the coverage of ontologies is maximised is by increasing its level of abstraction. It would thus seem that very general ontologies are ideal candidates for reuse. Nonetheless, a higher level of abstraction increases the gap between concept definitions in the reused ontology and the level at which new concepts are ideally defined. Reusing an abstract ontology means that the concepts it defines provide little direct structure for new concept definitions. It thus requires extra work to bridge this gap.

Lastly, if the ontology is specified in a suitable representation language, the inheritance of properties and other implicit knowledge can be used to check not only consistency but also the extra-logical quality of the ontology extension: whether what is derived about new classes and properties makes sense (Section 5.2). Note that a more formal representation language has only a partial effect on level of formality. An ontology in OWL 2 DL that specifies only named classes and subclass axioms is in fact only in the $\mathcal{AL}$ fragment of DL, whereas an ontology in the same language that uses some of its more expressive features may be in full $\mathcal{SROIQ}(D)$. Corresponding representation languages signal compatibility of inference – inferences on definitions in the reused ontology propagate to new definitions – whereas a higher level of formality increases the *number* of inferences, and thus usability, but also constrains coverage to only those domains compatible with its entailments.

### 5.4.1 Types of Ontologies: Limits to Reuse

Ontology reuse received quite a lot of attention, most notably in the construction of ontology libraries (Breuker and Van De Velde, 1994; van Heijst et al., 1997). Rather than by purpose – as in Chapter 4 – van Heijst et al. distinguishes ontology types by the level of abstraction of their content (See Figure 5.3). This

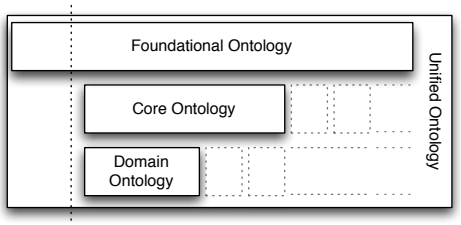


Figure 5.4: Contemporary ontology types

categorisation was based on the idea that the possibility for ontology reuse is influenced by the level of domain-dependence of the ontology.

van Heijst et al. (1997); Valente and Breuker (1996) distinguish five types of ontologies that provide a layered approach to reuse, and signal compatibility between knowledge based systems. A *top ontology* specifies highly generic categories and relations that are relevant to all domains. *Generic ontologies* specify general notions that are applicable across multiple domains, such as time, space, causality etc. A *core ontology* is a general ontology of application domains such as law, medicine and biology, and specifies a minimal set of concepts needed to describe entities in the domain. *Domain ontologies* specify the conceptualisation of a particular domain. Examples are the Linnaean taxonomy of species or an ontology of vehicles. An *application ontology* is an ontology that captures the ontological commitments of a single application. In this list, the representation ontology, of which the Frame Ontology is an example, is the odd-one out as it operates at a meta level with respect to concepts in an ontology (Section 4.4).

The distinction between these types is given by a theoretical consideration to organise the ontologies in an ontology library in a modular fashion and at strictly separate levels of abstraction. For instance, the core ontology type was initially proposed by van Heijst et al. (1997) as a means to better index domain ontologies stored in the library. Similarly, the specification of an application ontology is a fixed step in the *Common*KADS methodology for knowledge based systems development (Schreiber et al., 2000): a domain ontology is refined into an application specific ontology, which is then implemented in a knowledge based system. Although a similar modular organisation can be found in ONTOLINGUA, and more recently in the strict editorial process in OBO Foundry, the distinction is artificial. In practice ontologies rarely fit neatly into one of the categories. Contemporary ontologies can be roughly distinguished as *unified*, *foundational*, *core* or *domain* ontologies (Figure 5.4). Each of these ontology types holds a different position with respect to the trade off between the factors of coverage, abstractness and level of formality.

#### Unified Ontologies

Unified ontologies are ontologies that aim to present a 'grand unifying theory'; they cover both highly abstract as well as concrete concepts from a wide range of domains. Perhaps the most typical unified ontology is the SENSUS ontology of Swartout et al. (1996), which was developed explicitly with the purpose of reuse in mind. It is a large (50k) skeletal ontology with a broad coverage,

and contains both abstract and relatively concrete concepts from a variety of sources. Swartout et al. (1996) describe a general methodology where ontology construction should proceed by selecting and refining relevant parts of the SENSUS ontology. This way, it was thought, one could avoid "stovepipe" ontologies that can be extended vertically, i.e. along the specific-generic axis, but not horizontally, across domains.

However, formal definitions would impose interpretations on concepts that constrain their usability. For this reason, the ontology was relatively lightweight to maximise domain coverage, it is a knowledge management ontology. The Wordnet system of Miller (1990), which was in fact part of SENSUS, is often used in a similar way to populate ontologies. In the end, SENSUS did have some utility as source of inspiration for more specific ontologies, but could not enforce particular inference nor guarantee consistency between its different extensions. Another example of a unified ontology is the CYC ontology of Lenat et al. (1990); Lenat (1995), which is discussed in more detail in Section 6.2.1. In contrast to the SENSUS ontology, CYC is specified using an expressive formal language.

Concluding, unified ontologies are not the most suitable for direct reuse. When lightweight they provide little guidance in defining new concepts, and when rigourously formal their broad coverage introduces significant computational overhead.

#### Foundational Ontologies

Foundational ontologies or *upper* ontologies, are hybrid ontologies that combine top ontologies with the description level of a representation ontology. They have a philosophical disposition and focus on the basic categories of existence. Examples are SUO/SUMO (Niles and Pease, 2001), DOLCE (Masolo et al., 2003), BFO (Grenon, 2003) and the General Formal Ontology (Herre et al., 2006).[17] The foundational ontology ascribes to a role similar to that of unified ontologies in that it is aimed at coverage of a broad range of domains. Foundational ontologies find their roots in the philosophical discipline of formal Ontology.[18] In fact, most philosophical ontologies are foundational.

Contrary to unified ontologies, foundational ontologies are based on the stance that 'abstract is more': their applicability to a multitude of domains is ensured by defining general, abstract notions rather than by the direct inclusion of a large number of relatively specific concepts. Although the foundational approach ensures rigorous ontological definitions and broad coverage, this comes at a cost. First of all, it is to be expected that the rather abstract nature of foundational ontologies makes them less suitable candidates for knowledge acquisition support in the development of concrete domain ontologies. For instance, the developer of a domain ontology of pet animals has little to gain from such philosophically inspired classes as a:FiatObjectPart and a:GenericallyDependentContinuant in BFO. The distance between concepts in the foundational ontology and concrete domain concepts is just too big (cf. Figure 5.2).

[17] Note that although DOLCE was not initially developed as foundational ontology, it is often used as such. See Chapter 6.

[18] SUMO is an exception to the rule, as it only partly relies on the top ontology of Sowa (2000) and provides relatively concrete definitions in the physical domain.

A related issue is that reasoning over the top level categories in foundational ontologies may not necessarily have a direct bearing on practical reasoning problems because of a possible proliferation of inferences over abstract categories. Furthermore, their philosophical disposition makes that they are traditionally specified in a form of first order logic, and define meta-level categories at the ontological level of Guarino (1994). Although OWL compatible versions are sometimes available, the limited expressiveness of this language means that such versions are ontologically less strict, and are not entirely faithful to the full ontology.

#### Core and Domain Ontologies

Core ontologies balance coverage and abstractness by restricting the scope to a particular area of expertise. Examples of such ontologies are the LRI-Core and LKIF Core ontologies of Breuker and Hoekstra (2004a); Breuker (2004) and Breuker et al. (2007); Hoekstra et al. (2008, this volume) respectively, the Core Legal Ontology and Fishery Ontology of respectively Gangemi et al. (2005) and Gangemi et al. (2004), and more domain independent ontologies such as the COBRA ontology of business processes analysis (Pedrinaci et al., 2008), or for information integration (Doerr et al., 2003). Although the scope restriction limits the possible situations in which core ontologies can be used, it has several advantages. A core ontology is more likely to provide definitions of concrete concepts which are more intuitive candidates for reuse, and can sanction more relevant inference.

On the other hand, the concreteness of core ontologies is not a necessary given. Because important domains such as Law and Medicine are highly specialised and theoretical, they include very abstract concepts one does not readily find in generic ontologies. Examples are the legal ontologies of Mommers (2002); Lehmann (2003); Rubino et al. (2006) that reflect stances in legal theory and cover such abstract concepts as rights, duties, liability and legal causation (see also Chapter 6).

Domain ontologies extend core (or foundational) ontologies with more concrete categories that are of direct relevance to inference in a specific domain. To give an example, the Fishery Ontology of Gangemi et al. (2004) is an extension of the DOLCE ontology and defines such concepts as Fishing_Area, Aquatic_Organism and Catching_Method that can be refined to Aberdeen_Bank, Herring and Pelagic_Trawling in a domain ontology on North Sea fishing. Similarly, a domain ontology of criminal law would introduce subclasses of Crime specific to a particular jurisdiction and iterate specific felonies and misdemeanours such as Manslaughter and Petty_Theft.

### 5.4.2 Safe Reuse

It can be necessary to reduce the expressiveness of a reused ontology for it to fit a different, more restricted knowledge representation formalism. This practice is often seen in cases where foundational ontologies are applied in a Semantic Web context. For instance, in the SWIntO ontology of Oberle et al. (2007)[19], which is a combination of SUMO and DOLCE, the expressiveness of SUMO

[19] SWIntO: SmartWeb Integrated Ontology

is constrained by stripping it of most (if not all) of its axioms, leaving only a bare-bone subsumption hierarchy. However, this weakens the role of an ontology as source for ontological choices and design patterns as any modification of an ontology compromises the commitment of the ontology to its original conceptualisation.

To illustrate this point, consider an ontology as a set of ontological commitments. These commitments are comprised of a set of (non-logical) symbols representing the entities the ontology claims to exist, the *signature* of the ontology, and a set of *ontological statements* about these entities.[20] A difference in either of these sets of commitments makes a different claim about reality – or what can exist in a knowledge base – and thus denotes a different ontology.

Removing a symbol or statement means that what is reused is no longer the original ontology, but rather a scaled-down version. Because in that case, no direct match of the ontological commitments of the source ontology and the new ontology can be established, this practice compromises the purpose of ontology reuse. Namely, to ensure compatibility between the different ontologies that reuse and commit to the same, more general ontology (Gruber, 1994). This traditional reuse is only possible by full *formal* reuse as opposed to *informal reuse*:

**Definition 5.4.1 (Informal Reuse)** *The* informal reuse *of one ontology $\mathcal{O}'$ by another ontology $\mathcal{O}$ requires (at the least) some syntactic correspondence between the presence of named entities in the signatures of both ontologies: $\mathsf{Sig}(\mathcal{O}) \cap \mathsf{Sig}(\mathcal{O}') \neq \emptyset$.*

In other words, informal reuse does not imply a *formal* commitment to the theory expressed by the reused ontology: the commitment exists only with respect to part of the signature of the reused ontology, and cannot be established through formal means. Of course, informal reuse can range from a very loose correspondence to a tight integration. Informal reuse can still be quite rigorous and close to the original interpretation by a commitment to other than formal requirements, such as a specification in natural language.

For a more formal notion of reuse, we further restrict the definition of reuse by adopting an imports paradigm: the reusing ontology should 'import' all ontological commitments of the reused ontology. Ontology import is generally defined using two closure axioms (Motik et al., 2009):[21]

**Definition 5.4.2 (Imports Closure)** *The transitive* imports closure *of an ontology $\mathcal{O}$ consists of $\mathcal{O}$ and any ontology that $\mathcal{O}$ imports.*

**Definition 5.4.3 (Axiom Closure)** *The* axiom closure $\mathcal{O}_\cup$ *of some ontology $\mathcal{O}$ is defined as the smallest set that contains all axioms from each ontology $\mathcal{O}'$ in the* imports closure *of $\mathcal{O}$.*

These closures are syntactic constructs that gather together the set of all axioms, formal ontological statements, that express an ontology's commitments.

[20] Formally, the signature of an ontology is the set of all non-logical symbols, i.e. all entities in the ontology excluding the symbols of the knowledge representation language and sentences over the non-logical symbols. Note that ontologies specified in a non-formal language can be said to have a similar signature, albeit not as explicit.

[21] Though these definitions lie at the heart of OWL 2's import mechanism (Section 3.5.2), they are not necessarily particular to that language.

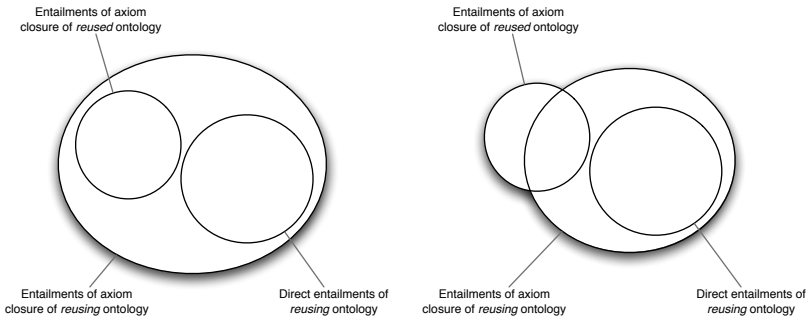


Figure 5.5: Relation between entailments in full and partial formal reuse

The partial *semi formal* reuse of e.g. SUMO by SWIntO, i.e. the commitment to just the taxonomic hierarchy of SUMO without considering its complex class axioms and properties, can be expressed in terms of these closures in a straightforward manner:

**Definition 5.4.4 (Semi Formal Reuse)** *An ontology $\mathcal{O}'$ is reused by ontology $\mathcal{O}$ through* partial semi formal reuse *if the intersection of the axiom closures $\mathcal{O}'_\cup$ and $\mathcal{O}_\cup$ is non empty:*

$$\mathcal{O}'_\cup \cap \mathcal{O}_\cup \neq \emptyset$$

Full semi formal reuse *is the case where the axiom closure $\mathcal{O}'_\cup$ is a subset of the axiom closure of $\mathcal{O}_\cup$:*

$$\mathcal{O}'_\cup \subseteq \mathcal{O}_\cup$$

Although this reuse ensures the inclusion in $\mathcal{O}$ of at least some axioms of the reused ontology $\mathcal{O}'$, it is only semi formal. Axioms are regarded as distinct *syntactic* entities, without taking their semantics into account. However, the axioms of different ontologies may interact, and cannot be considered separately. Even in the case of full semi formal reuse, combining previously unrelated axioms can have surprising effects.

When some ontology $\mathcal{Q}$ is imported by an ontology $\mathcal{P}$, the semantics of the resulting axiom closure $\mathcal{P}_\cup$ is determined by the union of the axioms $\mathcal{P}_\cup = \mathcal{P} \cup \mathcal{Q}_\cup$. Although $\mathcal{P}$ imports all axioms from $\mathcal{Q}_\cup$ this is no guarantee that the ontological commitments of $\mathcal{P}_\cup$ are compatible with that of $\mathcal{Q}_\cup$. The intention of reuse is that axioms in $\mathcal{P}$ interact with the axioms in $\mathcal{Q}_\cup$ in such a way that in $\mathcal{P}_\cup$ new entailments may hold for the axioms in $\mathcal{P}$. However, the meaning of terms in $\mathcal{Q}_\cup$ may change as a consequence of the import as well (Cuenca Grau et al., 2007).

For instance, consider the following axioms in $\mathcal{Q}$:

$$\begin{aligned} \mathsf{Bird} &\sqsubseteq \mathsf{has\_part}\ \textbf{exactly}\ 2\ \mathsf{Wing} \\ \mathsf{Penguin} &\sqsubseteq \mathsf{Bird} \end{aligned}$$

Because of the first axiom, we can infer $\mathsf{Penguin} \sqsubseteq \mathsf{has\_part}\ \textbf{exactly}\ 2\ \mathsf{Wing}$: because penguins are birds, they also have exactly two wings. Suppose that $\mathcal{P}$

imports $\mathcal{Q}$ and refines the definition of Bird with the axiom:

$$\mathsf{Bird} \sqsubseteq \mathsf{has\_ability}\ \textbf{some}\ \mathsf{Flying}$$

Thus, where $\mathcal{P}$ entails that birds can fly, $\mathcal{P}_\cup$ entails that birds can fly and have two wings. However, this seemingly innocent (and useful) extension will also allow us to infer that penguins can fly: $\mathsf{Penguin} \sqsubseteq \mathsf{has\_ability}\ \textbf{some}\ \mathsf{Flying}$. Without altering $\mathcal{Q}$ directly, adding a single axiom in $\mathcal{P}$ has changed the meaning of terms in the ontology: its ontological commitments in the context of $\mathcal{P}$ differ from its explicit commitments.

Of course, it is easy to come up with more far reaching modifications to $\mathcal{Q}$, such as adding disjointness axioms or meta modelling. This is problematic from a traditional reuse perspective, as some other ontology $\mathcal{P}'$ may reuse and extend $\mathcal{Q}$ as well, and introduce yet other interactions with the axioms of $\mathcal{Q}$:

**Definition 5.4.5 (Ontology Interaction Problem)** *The* ontology interaction problem *is the general problem that if an ontology reuses another, its ontological commitments may affect the commitments of the reused ontology and thereby undermine its compatibility with other ontologies that reuse the same ontology.*

How then to make sure that $\mathcal{P}$ and $\mathcal{P}'$ are compatible, and information exchange between two systems respectively committing to these ontologies will be possible? It is clear that because of the syntactic nature of the axiom closure, the union of two closures does not give us an orderly definition of reuse. At the least, formal reuse of an ontology $\mathcal{O}'$ by an ontology $\mathcal{O}$ should not affect the original entailments of $\mathcal{O}'$, and if any part $\mathcal{O}'' \subseteq \mathcal{O}'$ is (for some reason) left outside the axiom closure of $\mathcal{O}$, its entailments should neither be affected by $\mathcal{O}$ nor vice versa. In other words, any partial reuse of $\mathcal{O}'$ may only ignore that part $\mathcal{O}''$ that is wholly disconnected and irrelevant to the extension $\mathcal{O}$.

Ghilardi et al. (2006) introduce a notion of *safe reuse*, direct formal reuse of a signature in an ontology, based on the definition of *conservative extension* in description logics:

**Definition 5.4.6 (Conservative Extension)** *An ontology $\mathcal{O}$ is an* S*-conservative extension of $\mathcal{O}_1$, if $\mathcal{O}_1 \subseteq \mathcal{O}$ and for every axiom $\alpha$ expressed using signature* S*, we have $\mathcal{O} \models \alpha$ iff $\mathcal{O}_1 \models \alpha$. The ontology $\mathcal{O}$ is a* conservative extension *of $\mathcal{O}_1$ if $\mathcal{O}$ is an* S*-conservative extension of $\mathcal{O}_1$ with respect to the signature of $\mathcal{O}_1$.*

In other words, the axiom closure $\mathcal{P}_\cup$ is only a conservative extension of $\mathcal{Q}_\cup$ if no axioms are implied by $\mathcal{P}_\cup$ over symbols in the signature of $\mathcal{Q}_\cup$ that are not also implied by $\mathcal{Q}_\cup$ itself. This notion can be straightforwardly applied to establish that in our case, $\mathcal{P}_\cup$ is not conservative with respect to $\mathcal{Q}_\cup$ because in $\mathcal{P}_\cup$ penguins can fly, and in $\mathcal{Q}_\cup$ they cannot.

An ontology $\mathcal{O}$ is *safe for a signature* if no other ontologies $\mathcal{O}'$ using that signature can exist for which $\mathcal{O} \cup \mathcal{O}'$ is not a conservative extension of $\mathcal{O}'$ (Cuenca Grau et al., 2007). Unfortunately, Cuenca Grau et al. also show that determining whether some set of axioms (an ontology) is safe with respect to a signature is undecidable for expressive description logics.

It is clear that formal reuse of an ontology must be complete with respect to the logical consequences of axioms expressed in that ontology (Figure 5.5). But not only must the reusing ontology be consistent with the source ontology, it also should be a conservative extension. Note, however, that formal

ontology reuse does not require the existence of an imports relation between two ontologies, but rather ensures compatibility between their commitments. Nor does formal reuse necessarily require a compatibility between the representation formalisms used by both ontologies, rather determining whether an extension is conservative easily becomes undecidable for arbitrarily differing knowledge representation languages.

Recently, several tools were developed to support *partial formal reuse* by means of ontology module extraction Konev et al. (2008); Jimenez-Ruiz et al. (2008). In this case, some ontology may $\mathcal{O}$ reuse part of the axioms in an ontology $\mathcal{O}'$ provided that it is safe with respect to the signature of the reused axioms. Module extraction is a process by which exactly those axioms that affect the semantics of axioms in the reusing ontology are extracted to form a coherent whole. Of course, this is only of practical use when the reused ontology has limited density, i.e. when axioms in the ontology are relatively independent. Furthermore, the safety of $\mathcal{O}$ with respect to the signature of axioms from $\mathcal{O}'$ for expressive DLs can only be determined using *locality conditions* (Cuenca Grau et al., 2007; Jimenez-Ruiz et al., 2008), or for languages that have favourable properties (Konev et al., 2008).

### 5.4.3 Possibilities for Reuse

The previous section discusses the general problem of reusing knowledge expressed by some ontology. The original ideals of direct ontology reuse between systems expressed by Gruber (1994) and others, seem to be only attainable given the following restrictions:

**Restriction of Non-Modification**
Reused ontologies should not themselves be modified in any way.

**Restriction of Formal Reuse**
Ontological correspondence between systems can only be ensured if all ontologies are expressed in a formal knowledge representation language.

**Restriction of Compatibility**
Formal languages used to express the ontologies that share ontological commitments for the purpose of knowledge reuse, should be compatible in such a way that the entailments of one ontology can be checked for consistency with any of the other ontologies.

**Restriction of Complete Reuse**
A reusing ontology should directly import the ontological commitments of the (part of the) reused ontology it commits to, i.e. it should import both its signature and ontological statements.

**Restriction of Safe Reuse**
A reusing ontology should at least be a conservative extension of the reused ontology, and it should ideally be safe with respect to the signature of the axioms it reuses.

Given these restrictions, we have an extra, formal tool for assessing the suitability of ontologies for reuse that augments the more conceptual considerations discussed in Section 5.4.1, and the distinction between knowledge management, representation and formal ontologies of Chapter 4. Table 5.3 shows

| Type | Purpose | Level | Meta | Expressiveness |
|---|---|---|---|---|
| Unified | KM/KR | abstract/concrete | yes/mixed | FOL, informal, restricted |
| Foundational | FO | abstract | yes | FOL |
| Core | KR/FO | abstract/concrete | no | FOL, restricted |
| Domain | KM/KR | concrete | no | restricted |

Table 5.3: Characteristics of ontology types and purposes

some of the typical characteristics of the types of ontologies introduced in that section. Unified ontologies operate at both abstract and concrete levels of description, involve meta modelling and are expressed using languages of varying expressiveness. Foundational ontologies are limited to the definition of abstract notions using (variants of) first order logic (FOL) at a meta level. Some have versions expressed using a more restricted language. Core ontologies do not use meta modelling in the definition of concepts, and mix abstract and concrete levels of description. They are usually, but not always represented using the restricted expressiveness of knowledge representation languages. Domain ontologies offer the most concrete concept definitions in languages of restricted expressiveness.

The use of different formal languages reflects the difference between an abstract more philosophically inspired perspective and the concrete practical perspective of knowledge engineering. In Section 4.4 this distinction was already discussed in more detail. The necessity of formal reuse shows a discontinuity in the layered approach for reuse aspired to by Valente and Breuker (1996); van Heijst et al. (1997); Schreiber et al. (2000): abstract and concrete ontologies do not *connect*. Foundational ontologies can only be reused by ontologies at a concrete level if they use the same expressive formalism. However, expressive languages such as KIF and CommonLogic are generally undecidable and no formal nor pragmatic means to ensure safe reuse are available.[22] In part this is overcome by the availability of variants of foundational ontologies in restricted, decidable languages, but these do not capture the full set of ontological commitments of the original.

## 5.5 Ontological Principles

Besides being a jump start in the form of a conceptual coat rack, the reason for reusing some ontology during ontology construction is the inclusion and adoption of its ontological choices. If these choices are explicit in the form of ontological commitments of the reused ontology, they can be directly applied in the construction of new concept definitions. Where the reused ontology is sufficiently dense in its definitions, it may function as a mould that prevents the ontology engineer from inadvertently constructing ontological oddities. However, as discussed the previous section, the direct reuse of pre existing ontologies is subject to several limitations. Furthermore, although formal ontological commitments may make the ontological choices explicit, it can be quite difficult to extract, and reverse engineer these choices from a bag of axioms expressed using a complex formal language. It may therefore pay to improve the accessibility of ontological choices by making explicit some of the principles

[22] CommonLogic allows for the specification of decidable dialects.

underlying the construction of ontologies. These *ontological principles* can then be applied as extra methodological guidelines during ontology construction. Where the design criteria of Gruber (1994) pose requirements for the ontology as a whole, ontological principles provide guidance in the representation of an entity as belonging to a fundamental category, i.e. what *kind* of entity it is, or as to which aspects of that entity are of ontological relevance.

### 5.5.1 OntoClean

Perhaps the most influential set of ontological principles can be found in the ONTOCLEAN methodology of Guarino and Welty (2002, 2004). ONTOCLEAN is a methodology for evaluating "the ontological adequacy of taxonomic relationships" and can be applied both while building a new ontology, or in its post hoc evaluation. It aims to provide a formal framework and vocabulary for justifying why a particular way of structuring a taxonomy is better than another. Central to the ONTOCLEAN methodology are the notions of *essence* and *rigidity* on the one hand, and *identity* and *unity* on the other. Once these notions are applied to two classes, they can assist to determine whether a subsumption relation between the two classes may hold.[23]

Class membership of some entity is *essential* to that entity if the entity could not exist were it not for being an instance of that class. For example, it is essential for human beings to have a mind: every human being must always be an instance of the class of things that have a mind. In fact, whether the class membership of an entity is essential depends on membership of other classes. Note that in the example, essence is conditional on class membership of the human class. For some classes this dependency is stronger than for others, and reflects its *rigidity*. All entities that are member of a rigid class depend on that class for their existence. The class membership is essential to those entities, and they can only cease to be a member of the class by ceasing to exist. A rigid class has *only* members that depend on it in this way. For instance, a naive representation may hold that having a mind is rigid, but that would mean that we would hear a distinct metaphysical '*puff*' whenever someone lost his mind, metaphorically speaking that is. Instances of the human class, on the other hand, do rely heavily on being a human for upholding their existence and it is therefore more appropriate case of rigidity: it is essential for all humans to be human beings. The rigidity of a class is the consequence of an ontological choice by the ontology engineer and thus reflects an important ontological commitment.

Non rigid classes can be inessential to some of their instances, they are *semi-rigid*, or not essential to *any* of their instances. Classes of the latter category are *anti-rigid*, typical examples of which are *roles* and *functions* such as the classes student and hammer: there is not a student that ceases to exist upon graduation. Here again, the example seems obvious, but the developer of an ontology for a library system may well want to automatically terminate library membership for graduates: a case where the rigidity of studentship is defendable.

A useful perspective to determine the rigidity of some class is the degree to which class membership is ontologically *subjective* or *objective* (Searle, 1995,

[23] The ONTOCLEAN papers use the logico-philosophical meaning of the term 'property' throughout. As this meaning differs substantially from the way in which the term is used in knowledge representation languages, the description of ONTOCLEAN will use standard KR vocabulary.

and Section 7.3). Where the existence of subjective entities depends on being felt by subjects (i.e. people), ontologically objective entities exist independently of any perceiver or mental state. This distinction helps to recognise features *intrinsic* to nature, and those *relative to the intentionality* of observers. Describing an object as a hammer specifies a feature of the object that is *observer relative*, whereas the existence of the physical object does not depend on any attitudes we may take toward it. Observer relative features are therefore ontologically subjective, and classes defined by such features are generally anti-rigid.

Deciding whether a class is essential, rigid, semi-rigid or anti-rigid in the OntoClean methodology thus puts the burden on the ontology engineer to contemplate all possible uses of the ontology. According to the methodology, the ontological commitments should be made explicit by marking elements in the ontology by means of a meta-property. This meta-property can then signal allowed use of the classes defined in the ontology. One restriction, for instance, is that rigid classes cannot be the subclass of anti-rigid classes and vice versa: the student class is therefore not a subclass of human.

Another important notion in the methodology is that of *identity*: how can we establish whether two entities are the same? Equality of entities is determined by means of *identity criteria*, and conversely if equality between entities is asserted, the identity criteria must necessarily hold. At first sight this seems very similar to the notion of essence, e.g. how in DL individuals are determined to belong to a class (See a.o. Section 7.2). However, the emphasis here lies on the properties that two individuals must have for them to be the *same individual*, e.g. for an owl:sameAs relation to hold between them. Guarino and Welty (2004) give the example of the relation between a duration and interval class. A sentence such as "All time intervals are time durations" does not entail that duration subsumes interval, as two intervals that have the same length are not necessarily the same, whereas two durations with the same length are. Although the same conclusion can be reached by giving a more conceptual account, i.e. the duration *is* the length of the interval, explicitly marking properties of classes as carrying identity criteria can be very useful to ensure innocuous reuse.

An example of an identity criterion is the inverse functional property type of OWL, which allows us to infer that two individuals are the same when they are related to the same individual via that property. For instance, two individuals that have the same social security number, must be the same person. Although a knowledge representation formalism may provide rather little in the way of built-in constructs for expressing identity criteria, these criteria can still be very useful in determining, or rather disqualifying, subsumption relations between classes.

The third notion, *unity*, refers to the property of entities as being wholes that are composed of parts (that may themselves be wholes). Unity can be used to differentiate different classes based on their mereological properties. For instance, water does not generally have parts (it carries *anti-unity*), whereas oceans do. As a rule, categories referred to using mass nouns do not carry unity, and categories that do are named by count nouns. Unity can be ensured by any of the traditional types of mereological relations, such as functional decomposition, spatial configuration, topology etc.. Subsumption between classes is valid only if their unity criteria are compatible: whether a class carries unity, anti-unity or no unity is inherited to its subclasses.

#### Formal Evaluation

The ONTOCLEAN methodology provides a formal framework for evaluating ontologies. It is instrumental in the formal elicitation of the ontological decisions underlying subsumption relations. Annotating classes with meta properties allows for automatic evaluation of the taxonomic relations in the ontology, i.e. if a class C subsumes C′ then:

- If C is *anti-rigid*, then C′ is anti-rigid: all students, be they lazy or overzealous can graduate without ceasing to exist.
- If C carries some identity criterion, then C′ must carry the same criterion: if organisms can be uniquely identified by their DNA, then any homo sapiens, canis familiaris, limax maximus and amoeba proteus can be as well.
- If C carries some unity criterion, then C′ must carry the same unity criterion: if a table consists of physical parts that enable its function in providing a level surface on which objects can be placed, then all tables do.
- If C has *anti-unity*, then C′ must also have anti-unity: if water does not have parts, then potable water doesn't either.

Meta properties are not just helpful when evaluating an ontology or constructing a new ontology. They can be used to put additional constraints on the validity of ontology reuse, provided that both the new and the reused ontology are annotated. However, the annotations are meta statements over axioms in the ontology and are not part of their semantics. Though formal, they cannot be used in the same way as the definition for conservative extensions to solve the ontology interaction problem (definitions 5.4.6 and 5.4.5, respectively). Rector (2003) recognised the two approaches as complementary parts of a two-step normalisation: ontological and implementation normalisation.

A limitation of the ONTOCLEAN methodology is that the distinctions it advocates, though inspired by philosophy, are an abstraction of established practices in knowledge engineering. For instance, the notion of anti-rigidity reflects a meta-theory of the distinction between roles and role-players (Steimann, 2000; Masolo et al., 2004), properties that are always *attributed* to some entity (cf. Section 6.3.2, Section 7.3.2). They reflect things an entity can *have* or *play*. Similarly, the notion of unity stands in direct correspondence to the distinction between objects and the substances they are made of.

Guarino and Welty are careful to avoid presenting ONTOCLEAN as a traditional methodology that provide guidelines for *taking* modelling decisions, rather the meta-properties are purely metadata that *signal* a particular ontological position with respect to some category. Though making these meta-properties explicit helps to communicate design decisions, the rationale for making these decisions is lost. In fact, because these meta-properties are derived from more concrete distinctions it begs the question whether the ONTOCLEAN approach gives more solid footing for an ontology engineer than a concrete guideline would.

Furthermore, that meta properties are enforced down the subsumption hierarchy holds for other properties as well, and is provided 'for free' by any serious knowledge representation language. That an ontology engineer needs to take this inheritance into account is indeed a source of many modelling errors, but is not particular to the meta-properties of the OntoClean approach.

### 5.5.2 Frameworks and Ontologies

The OntoClean methodology sprouted from a need to provide a means to evaluate the correctness of taxonomic relations, but is silent where it concerns the appropriateness of categories included in the ontology. In ontology representation languages such as OWL, the distinction between 'ontology' and actual situations coincides with the contrast between terminological knowledge and assertional knowledge: classes in the TBox and individuals in the ABox. However, this distinction is not concise enough to separate proper ontological categories from other, general terminological knowledge. In fact, not all concepts are suited for inclusion in an ontology, and in several papers we advocate a distinction between ontologies and so-called *frameworks* (Breuker and Hoekstra, 2004c; Breuker et al., 2007; Hoekstra et al., 2007, 2008).

As a rule, terminological knowledge is generic knowledge while assertional knowledge describes some state of affairs. These states can be generalised to patterns typical to particular kinds of situations. To be sure, if some pattern reoccurs and has a justifiable structure, it might evidently pay to store this structure as generic description. For instance, it may capture a predictable course of events. The combination of the situations and events related to eating in a restaurant is a typical example, and served in the Seventies to illustrate the notion of knowledge represented by scripts or 'frames' (Schank and Abelson, 1977; Minsky, 1975, respectively). A representation of this kind of generic knowledge, which is indeed terminological, is not an ontology.

Frameworks have a different structure than ontologies and capture systematic co-occurrence of the structural *relations* something has with other things. They describe such things as how activities are causally or intentionally related, or how objects are spatially and functionally configured: frameworks are primarily defined through *unity criteria*. Ontologies, on the other hand, define what things *are* in and of themselves and emphasise the essential, intrinsic properties of entities. Arguably, at a formal level the two are indistinguishable: every class in OWL is defined in relation to other classes, it cannot be otherwise. The distinction between frameworks and ontologies is therefore conceptual and does not coincide with the attributes of representation formalisms. For this reason, to keep an ontology free from contextual knowledge, we need to distinguish between those relations that make what a concept *is* and relations that place a concept in a particular frame of reference. For sure, the ontological definition of a concept is dependent on its context within the structure of the ontology, but it should be impartial to the context of its *instances*.

Take for example a *hammer*, its composition of head and shaft is not by accident: it is this particular combination that allows the hammer to be used 'as a hammer'. However, the mereological relation between the hammer as object, and its composites is not part of its ontological definition. Many different kinds of hammers exist, e.g. sledgehammers, mallets, conveniently shaped stones etc., each of which differs ever so slightly in the nature of its composites

and the relations between them. But they are all hammers. It is therefore only the function of the hammer as an instrument that defines what a hammer *is*, its mereological properties are merely circumstantial evidence. Searle (1995) stressed that the hammer-as-such cannot be substituted with the hammer-as-object; whether some object is a hammer is a social fact and depends on an attribution of the hammer function given a context of use. This attribution pinpoints exactly the conceptual difference between ontologies and frameworks. Where frameworks may incorporate the context in which this attribution holds by default and thus (over) generalise the typical physical features of a hammer-as-object to the hammer-as-such, ontologies should maintain an explicit distinction between the two.

From a methodological point of view, this allows us to introduce a rule of thumb: a combination of concepts and relations, as in e.g. a class restriction, is only to be considered part of an ontology when this particular combination is either systematic and independent of context, or when it makes the context explicit. A possible second consideration is inspired by the limitations imposed on admissible structures by the knowledge representation formalism. Of course this is a rather practical consideration, but nonetheless useful when considering the widespread use of OWL. If the primary task of ontology is to describe what things are, then a representation formalism specifically tailored to classification can be a fair benchmark for determining whether the kind of knowledge you want to express should be considered part of ontology or not. Given the limited expressiveness of OWL DL, especially because of its tree-model property (Motik et al., 2005), many frameworks are more easily represented using a rule formalism as they often require the use of variables. In fact, as we show in Hoekstra and Breuker (2008) and Section 7.2, OWL can be used to represent non-tree like structures commonly found in frameworks but only in a very round-about and non-intuitive manner. Furthermore, *epistemological frameworks* may define epistemic roles which can only be applied by reasoning architectures that go beyond the services provided by OWL DL reasoners (e.g. when they require meta-level reasoning). The limitations of OWL thus indicate a correlation between the conceptual distinction and representation formalisms. However, the two perspectives should not be confounded. Frameworks belong to the T-Box of any knowledge representation system, independent of whether it is based on a DL formalism or not.

We can distinguish three kinds of frameworks:

#### Situational frameworks

Situational frameworks are most characteristic for the notion of framework in general, because of the strong emphasis on context and teleology they share with frames and scripts (Minsky, 1975; Schank and Abelson, 1977). They are the stereotypical structures we use as plans for achieving our goals given some recurrent context, such as making coffee. These plans are not necessarily private to a single agent, and may involve transactions in which more than one actor participates. For instance, the definition of Eating-in-a-restaurant[24] is a structure consisting mainly of dependencies between the actions of clients and service personnel.

[24] In the following all concepts will start with a capital letter, properties and relations will not

Another notable characteristic of situational frameworks is that they are not subclasses of the goal directed actions they describe. For instance, Eating-in-a-restaurant is not a *natural* sub-class of Eating but rather refers to a typical model of this action in the situational context of a restaurant. Furthermore, it usually does not make sense to define subclass relations between situational frameworks themselves. Although we can easily envisage a proliferation of all possible contexts of eating – Eating-at-home, Eating-with-family, etc. — but does eating in a French restaurant fundamentally differ from eating in a restaurant in general? (Bodenreider et al., 2004; Breuker and Hoekstra, 2004a).

From a legal perspective, situational frameworks can be imposed on actual situations through articles of procedural ('formal') law. Although the *stereotypical* plans given in by custom, and the *prescribed* plans of law differ in their justification – rationality vs. authority – their representation is largely analogous. Similarly, legal norms combine generic situation descriptions with some specific state or action, where the description is qualified by a deontic term. For instance, the norm that "vehicles should keep to the right of the road" states that the situation in which a vehicle keeps to the right is obliged.

In short, situational frameworks are a fundamental part of the way in which we makes sense of the world, and play a prominent role in problem solving.

#### Mereological frameworks

Most entities, and objects and processes in particular, can be decomposed into several parts: they are *composites*. As we have seen in the hammer example, it can be tempting to incorporate a mereological view in the definition of a concept. A typical example is the definition of Car as having at least three, and usually four wheels, and at least one motor. However, a full *structural* description of a concept's parts and connections goes beyond what it *essentially* is. Cars are designed with a specific *function* in mind, and although there are certainly some constraints on its physical properties relevant to its definition, these are limited to those constraints actually necessary for fulfilling that function. Concept descriptions that do iterate over all or most parts of the concept are *mereological frameworks*, and can appear under a large diversity of names: structural models, configurations, designs, etc. Mereological frameworks play a major role in qualitative reasoning systems (see e.g. Davis (1984); Hamscher et al. (1992)).

Arguably, the line between the mereological framework and ontological definition of a term is sometimes very thin. For instance, if we want to describe a bicycle as distinct from a tricycle, it is necessary to use the cardinality of the wheels as defining properties as these are *central* to the nature of the bicycle. On the other hand, the number of branches a tree might have hardly provides any information as to what a tree *is*.

#### Epistemological frameworks

Inference in knowledge based reasoning does not happen in isolation. It is part of a larger structure of interdependencies between steps in a reasoning process: arriving at new information, given some initial situation. Epistemology is the study of how valid knowledge can be obtained from facts. Generic description

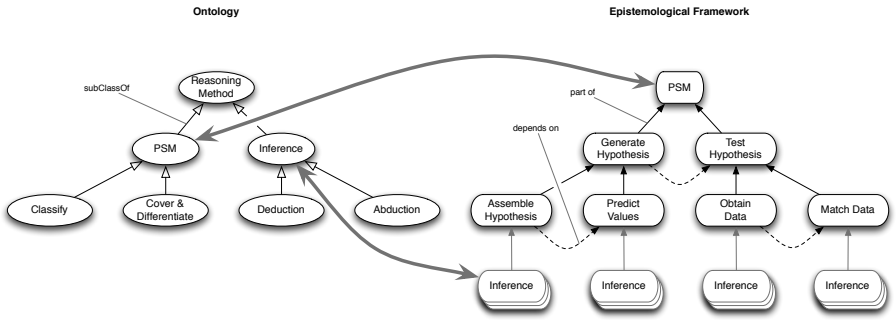


Figure 5.6: Ontology of reasoning terms vs. a framework for reasoning (Breuker and Hoekstra, 2004a)

of reasoning processes are thus *epistemological frameworks* that focus on the epistemological status of knowledge: the *use* of knowledge in reasoning, e.g. as hypothesis or conclusion (Breuker and Hoekstra, 2004c).

What sets the epistemological framework apart from mereological frameworks is its characteristic specification of dependencies between the distinct steps in a reasoning methodology. A typical example are the problem solving methods (PSM) discussed in Section 2.4.1 that are found in libraries of problem solving components (Breuker and Van De Velde, 1994; Motta, 1999; Schreiber et al., 2000). For instance, a problem solving method is not just a break-down of the mereological structure of some problem, but specifies control over the inferences it contains, such as the assessment of success or failure in attaining a sub goal. They invariably have at least two components: some method for selecting or generating potential solutions (hypotheses), and a method for testing whether the solutions hold. Whether a solution holds depends on whether it satisfies all requirements, or whether it corresponds with, *explains*, empirical data.

Figure 5.6 shows how an ontology of problem solving differs in perspective from a problem solving method described as epistemological framework. Although the concepts *inference* and *PSM* occur in both, the structure in which they are placed is decidedly different, as is also demonstrated by their sub categories. The epistemological framework has a mereological structure combined with sequential dependencies between its parts, the ontology is a subsumption hierarchy. To give a further illustration of the difference, consider the following action $a$:

> "Colonel Mustard kills Dr. Black in the conservatory, with the candlestick."

In a game of Cluedo $a$ will typically occur first as *hypothesis*, before it may or may not be accepted as *conclusion* (Colonel Mustard actually *did* kill Dr. Black). Being a hypothesis is therefore not essential to $a$, and in fact it never is: the hypothesis class is anti-rigid. Conversely the content of a hypothesis is not ontologically relevant to the concept itself; that $a$ is a different hypothesis from say "The candlestick has a huge dent in it" is only particular to the two individual hypotheses, but does not change what a hypothesis *is*. Similar to the Eating-in-

a-restaurant situation, the relevance of an $\alpha$-hypothesis class is dependent on the epistemological context of a particular problem: whodunnit?

**Ontology versus Epistemology** In general, the categories of epistemological frameworks can be defined in ontologies, provided that their definition is ontological: in Chapter 2 we have seen ample reason for keeping a strict separation between task dependent knowledge and domain knowledge, cf. the interaction problem of Bylander and Chandrasekaran (1987). Nonetheless, Bodenreider et al. (2004) discuss several other cases where (medical) ontologies are interspersed with epistemological categories of a different kind:

- Terms containing *classification criteria*, i.e. categories distinguished according to the way information is gathered to classify a particular instance as belonging to the category, e.g. *Murderer, convicted by jury trial* versus *Murderer, convicted by bench trial*. Although such information may be useful in justifying the conclusions of a knowledge based system, it does not constitute an ontological category.
- Terms reflecting *detectability*, *modality* and *vagueness*, e.g. *asymptomatic* diseases, that are detected without the patient displaying its symptoms. This information does not constitute a new ontological category, i.e. the treatment of an asymptomatic disease is the same as for symptomatic diseases.
- Terms created in order to obtain a *complete partition* of a given domain, such as *car*, *lorry*, *bicycle*, *other vehicles*. Often such bin-concepts are introduced because of a database perspective, where all categories need to be explicitly named.
- Terms reflecting mere *fiat boundaries*, e.g. a person of *average height* will have a different height in Asia than in Europe.

From a philosophical perspective the practice of having epistemologically loaded categories is problematic as these are decidedly non-ontological and do not correspond to categories of entities in reality. Knowledge representation ontologies are less scrupulous with regard to the ontological status of their definitions, and often some concessions will need to be made to facilitate the implementation of the ontology in a practical implementation (e.g. in application ontologies). However, epistemological promiscuity of categories is problematic as they are defined within the context of some shared epistemological framework or task. For instance, a category of asymptomatic diseases is only of relevance in the context of medical diagnosis, not in their treatment. And different trial procedures are mainly relevant when comparing different legal systems, but not when establishing a sentence.

Alles Vergängliche ist nur ein Gleichnis

*J.W. Goethe, Faust II, 1832*

## 5.6 Design Patterns

Most of the requirements and principles of the preceding sections pertain to what van Heijst et al. (1997) called *knowledge modelling* (Section 2.4.1): they are largely conceptual and independent of the language in which the ontology is eventually specified. However, a formally specified ontology is a *design model*, where the design is subject to *design principles* that guide the expression of the conceptual model in a formal language. As pointed out in Section 4.2, the language of choice for knowledge engineering ontologies that are to be shared on the web is the tailor made description logics formalism of OWL DL.

Knowledge modelling and design are not independent. Firstly, the choice of a representation language, and in particular its expressiveness, determines which (kinds of) ontologies can be reused. And secondly, safe reuse cannot be defined without reference to the language in which either ontology is represented. In both cases, the expressiveness of the ontology representation language is key, and in fact has a significant impact on what parts of the ontology as conceptual model can be formalised. In particular, the trade-off between expressiveness on the one hand and decidability and computational efficiency on the other played an important role in the specification of OWL DL (Levesque and Brachman, 1987, and Section 4.2). This technical trade-off is reflected by an ontological trade-off:

- not all ontologically relevant features can always be represented in OWL DL, and
- some features can be represented in multiple ways, each with its own benefits.

The encoding of an ontology is thus not just the result of several ontological choices, but also of design decisions. The requirements we reviewed so far do not directly support this process. For sure, it is the reuse of pre existing ontologies that lifts some of this burden. Not only does it provide off-the-shelve concepts and properties, but these convey insight in the decisions underlying the design of the ontology as well. The parallel with ontological choices has not gone unnoticed, and akin to the ontological principles of the previous section, the design decisions underlying several ontologies can be extracted as ontology *design patterns*:

**Definition 5.6.1 (Design Pattern)** *A* design pattern *is an archetypical solution to a design problem given a certain context.*

### 5.6.1 Patterns and Metaphors

Ontological definitions of concepts in a formal language do not differ much from descriptions of terms in natural language. In both cases an expression is constructed by combining symbols according to grammatical rules. The role of design patterns in ontologies can be made clear by analogy to the use of grammatical patterns in meaning construction. In fact, the way in which we combine words in linguistic expressions hints at the existence of several fundamental concepts in thought, and follows basic, cognitive rules (Pinker, 2007). One of the examples Pinker gives to support this hypothesis is the atypical behaviour of some verbs in the dative form. While most transitive verbs can be used both in a propositional dative form (subject-verb-'to'-recipient) and a double-object form (subject-verb-recipient-thing), this cannot be extracted to a general rule. For instance, whereas we can say:

- 'Give a muffin to a moose'
- 'Give a moose a muffin'

we cannot say:

- 'Biff drove the car to Chicago'
- 'Biff drove Chicago the car'

or, alternatively:

- 'Sal gave Jason a headache'
- 'Sal gave a headache to Jason'

Idiosyncrasies such as this indicate that the two constructions are not synonymous, but in fact follow differing underlying patterns. Whereas the propositional dative matches the pattern "cause to go", as in '*cause a muffin to go to a moose*', the double-object dative matches "cause to have", as in '*cause a moose to have a muffin*'. A plethora of other constructions such as these exists. For example, to indicate a distinction between direct and indirect causation as in '*dimming the lights*' when sliding a switch and '*making the lights dim*' when turning on the toaster. It is these patterns that are used to construct metaphors such as the *container* metaphor of Section 5.6.3 or a *conduit* metaphor where ideas are things, knowing is having, communicating is sending and language is the package:

> We *gather* our ideas *put* them *into* words, and if our verbiage is not *empty* or *hollow*, we might *get* these ideas *across* to a listener, who can *unpack* our words to *extract* their *content*.
>
> (Pinker, 2007, p.60)

According to Pinker, the basic concepts in a language of thought correspond to the kinds of concepts that fit the slots of grammatical constructions. The grammar rules of language reflect the structure of our conceptualisation of the world around us. It shows the restrictions on the ways in which we

can combine concepts to create meaningful categories. This is not the place to discuss whether or not Pinker's basic concepts are ontologically relevant (cf. Chapter 6), rather what is important here is that the position of some term in a sentence is indicative of its meaning, and the general category it belongs to.

We have seen that linguistic expressions follow patterns that can be re-applied to new circumstances to create new meaning. These patterns are design patterns, and depending on *which* pattern we apply, we create a different meaning or shift emphasis. However, not every pattern is applicable to just any term. In Pinker's example, the applicability of some grammatical pattern to a term is determined by an (in)compatibility between that which a pattern is meant to express about it, and the meaning of the word. As we will see, design patterns in knowledge representation and ontology development pose similar restrictions

### 5.6.2 Patterns in Ontology Engineering

Presutti et al. (2008) give a comprehensive overview of design patterns in ontology engineering and discuss a wide range of patterns. Though related to ontology engineering, not all of these capture design decisions. For instance, the definition is used to specify presentation patterns (naming conventions), documentation best practices (annotating classes), patterns for ontology decomposition, mappings between ontology vocabularies (e.g. SKOS to OWL), normalisation macros (asserting inferred information), and even grammar rules for translating natural language sentences into OWL axioms.

Their definition of ontology design pattern is influenced by its epistemological role as data structure in the specification of design patterns in an online library. For this application, it is useful to view a engineering design pattern as reification of the relation between a design schema (and its elements), and its possible implementations (and their elements):

**Definition 5.6.2 (Ontology Engineering Design Pattern (Presutti et al., 2008))** *An* ontology design pattern *is a modelling solution to solve a recurrent ontology design problem. It is an* information object *that expresses a* design pattern schema *or* skin*. A schema can only be satisfied by a* design solution*, which is a particular combination of ontology elements that fulfil certain* element roles *defined by the design pattern.*

Using this definition, patterns and their solutions can be properly indexed and documented. Furthermore, the definition is cast in terms of the DOLCE Ultra light ontology and the Descriptions & Situations extension of Gangemi and Mika (2003), which gives it a grounding in a larger foundational ontology. Nonetheless, the fact that the definition presents a design pattern as a reification obscures the reason *why* a solution 'satisfies' a schema. Especially since the ontological relevance of an individual design pattern relies on this relation, a proper definition should specify how this relevance can be assessed. This assessment should not rely on an explicitly asserted relation with concrete implementations. It seems that ontologically relevant design patterns are *themselves* the generic 'pieces' of a formal specification that can be applied, or implemented by an ontology.

### 5.6.3 Knowledge Patterns

Design patterns are not just generic descriptions that can be inherited over a subsumption hierarchy (Clark et al., 2000). The application of a design pattern in a concrete case is a source of inference when it applies to multiple aspects of the case. Clark et al. give a logic programming example of a container pattern that is used to express the capacity of a computer: "the computer contains memory" Arguably the term 'capacity' is ambiguous and may refer to a variety of measurable attributes of the computer, e.g. internal memory, hard disk space, or the number of expansion slots. Querying a DL knowledge base that applies the pattern in a straightforward way will return all possible capacities.

One solution is to *parameterise* the pattern by requiring the capacity to be of a certain type. Clark et al. reject this solution on the grounds that parsimony may be even further compromised when more distinctions are necessary, such as between physical and metaphysical containment ("the computer contains information"). In short, to ensure applicability to a broad range of cases a design pattern will need to explicitly include all possible dimensions along which its cases may differ. Such patterns will be overly complex and include a domain dependent bias: they cannot be applied freely to new cases.

The conclusion is that although it may seem that a subsumption relation holds between the container and a computer, this is incorrect. Rather, a computer can be *modelled as* a container: we apply a container metaphor. Clark et al. (2000) define a *knowledge pattern* as a theory whose axioms are not part of the knowledge base that implements it. The implementation of such a pattern occurs by importing its axioms, and mapping the symbols of the implementation to symbols in the pattern's signature. This mapping is given by a *morphism* that maps every non-logical symbol in the pattern to a corresponding symbol in the knowledge base. For instance, we can define Free_Space as that capacity of a Container that is not occupied:

$$\begin{aligned} \mathsf{Free\_Space} &\sqsubseteq \mathsf{Capacity} \sqcap \mathsf{capacity\_of}\ \textbf{some}\ \mathsf{Container} \sqcap \textbf{not}\ \mathsf{Occupied} \\ \mathsf{Container} &\sqsubseteq \mathsf{free\_space}\ \textbf{some}\ \mathsf{Free\_Space} \\ \mathsf{free\_space} &\sqsubseteq \mathsf{capacity\_of}^{-} \end{aligned}$$

This allows us to query the value of the free_space property to retrieve the free space of any container. By applying the morphism $m$:

$$\begin{aligned} m = \{&(\mathsf{Container}, \mathsf{Computer}), (\mathsf{Free\_Space}, \mathsf{Available\_RAM}), \\ &(\mathsf{Capacity}, \mathsf{RAM\_Size}), (\mathsf{Occupied}, \mathsf{Used\_RAM}), \\ &(\mathsf{free\_space}, \mathsf{available\_ram})\} \end{aligned}$$

we can construct the analogous definitions for Available_RAM and Computer:

$$\begin{aligned} \mathsf{Available\_RAM} &\sqsubseteq \mathsf{RAM\_Size} \sqcap \mathsf{capacity\_of}\ \textbf{some}\ \mathsf{Computer} \sqcap \textbf{not}\ \mathsf{Used\_Ram} \\ \mathsf{Computer} &\sqsubseteq \mathsf{available\_ram}\ \textbf{some}\ \mathsf{Available\_RAM} \\ \mathsf{available\_ram} &\sqsubseteq \mathsf{capacity\_of}^{-} \end{aligned}$$

Similar morphisms can be defined that map e.g. Free_Space onto Free_Slots or Free_Harddisk_Space etc.

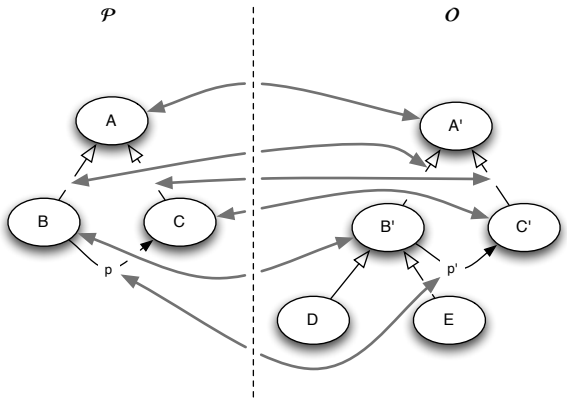


Figure 5.7: The mapping between ontology design pattern $\mathcal{P}$ and its implementation in ontology $\mathcal{O}$

The definition of knowledge pattern circumvents the limitations of pattern implementation based solely on subsumption, and introduces a weaker mapping function to connect patterns to their implementation. Nonetheless, knowledge patterns are rather restrictive in that both the pattern and the mapping are required to be external to the knowledge base.

### 5.6.4 Ontology Design Patterns

Viewing ontology design patterns as knowledge patterns allows for a formal evaluation of pattern implementation in ontologies. In this section I formalise and extend the most salient patterns of Presutti et al. (2008), and scrutinise them with respect to the reuse criteria discussed in Section 5.4.2. I define an ontology design pattern as a knowledge pattern that captures an ontological theory and can be implemented in other ontologies:

**Definition 5.6.3 (Ontology Design Pattern)** *An* ontology design pattern *is an ontology $\mathcal{P}$ that is said to be* implemented *in an ontology $\mathcal{O}$ iff a mapping function $M$ specified by a* signature morphism *exists between the signatures $\mathsf{Sig}(\mathcal{P})$ and $\mathsf{Sig}(\mathcal{O})$ that for all axioms $\alpha \in \mathcal{P}$ it holds that $\mathcal{O} \models M(\mathsf{Sig}(\alpha), \alpha)$, where $M(\mathsf{Sig}(\alpha), \alpha)$ is the set of axioms produced by applying $M$ to $\alpha$ and its signature.*

Figure 5.7 shows how a design pattern $\mathcal{P}$ is mapped onto ontology $\mathcal{O}$. Note however, that like knowledge patterns, this definition does not strictly require a bijective mapping between pattern and implementation. A symbol from the signature of $\mathcal{O}$ may occur an arbitrary number of times in the mapping $M$, and not all symbols from $\mathcal{O}$ must be mapped onto the signature of $\mathcal{P}$ (although all those from $\mathcal{P}$ must). Furthermore, it can be useful to weaken the definition of knowledge pattern to allow the mapping to be defined as axioms in $\mathcal{O}$. This way, the implementation relation does neither require nor preclude the use of subsumption relations to map the pattern ontology $\mathcal{P}$ to relevant parts of $\mathcal{O}$. Although this leaves room for some of the ambiguities Clark et al.'s definition is meant to circumvent, it allows for a straightforward implementation of the pattern in cases where the ambiguity does not hurt.

Presutti et al. (2008) distinguish *content* patterns and *logical* design patterns. Content patterns are in fact small ontologies, the reusability of which is advanced by explicit documentation of their design rationale and best practices. Gangemi (2005); Presutti et al. (2008) propose the development of a library of such content patterns that can be used as building blocks in ontology design. Logical design patterns are specified at a meta-level and implemented by *instantiation*. Axioms in the implementation are typed according to the meta-categories defined by the pattern.

Content patterns are defined as restriction on the knowledge pattern of Clark et al. (2000), on which definition 5.6.3 is based. Rather than a free mapping relation between pattern and ontology, an implementation of a content pattern should preserve *downward taxonomic ordering*. The signature of the content pattern is mapped only to symbols that are subsumed by symbols of the pattern. Presutti et al. (2008) do not give formal definitions of pattern implementation, but the gist of their proposal can be captured by an adaptation of definition 5.6.3:

**Definition 5.6.4 (Content Pattern)** *A* content ontology design pattern *is an ontology $\mathcal{P}_c$ that is said to be* implemented *in an ontology $\mathcal{O}$ iff $\mathcal{P}_c$ is implemented as ontology design pattern in $\mathcal{O}$, and if $\mathcal{P}_c$ were in its axiom closure (i.e. $\mathcal{P}_c \subseteq \mathcal{O}_\cup$) then for all axioms $\alpha \in \mathcal{P}_c$ it would additionally hold that $M(\mathit{Sig}(\alpha), \alpha) \sqsubseteq \alpha$ in $\mathcal{O}$.*

In other words, the axioms in $\mathcal{O}$ returned by the mapping function $M$ applied to $\alpha$ and its signature must be subsumed by $\alpha$.

If an ontology design pattern is itself an ontology, this raises the important question what its ontological status is. For instance, should a naive implementation of a pattern were $\mathcal{P}$ is imported and extended by $\mathcal{O}$ using standard OWL constructs meet the same safety requirements as normal ontology reuse? It is clear that the additional restriction on the implementation of a pattern posed by definition 5.6.4 makes that, though actual reuse of the pattern *as ontology* is not required, it still must be a conservative extension (definition 5.4.6). Consequently we can apply the criteria of a.o. Ghilardi et al. (2006); Cuenca Grau et al. (2007) to the union $\mathcal{P}_c \cup \mathcal{O}_\cup$ to determine whether $\mathcal{O}$ is a *safe implementation* of content pattern $\mathcal{P}_c$.

Because of their meta level character, logical design patterns are a different breed and have a more structural character. In Presutti et al. (2008), logical patterns are logical meta theories. To illustrate this, consider the top level categories of OWL itself. These are the language constructs that enable the specification of *any* valid OWL ontology. Logical patterns work in the same way; they define the meta classes and reified relations that specify valid implementations of the pattern. To give a formal definition:

**Definition 5.6.5 (Logical Pattern)** *A* logical pattern *is an ontology $\mathcal{P}_l$ that is said to be* implemented *in an ontology $\mathcal{O}$ iff there exists a set of axioms $\mathcal{Q}$ such that $\mathcal{Q} \subseteq \mathcal{O}$ and it holds that $\mathcal{P}_l \models \mathcal{Q}$.*

The part of an ontology that implements a logical pattern should be a valid model of the pattern. Logical patterns expressed in OWL cannot be enforced using a description logics classifier; they are limited to OWL Full (cf. Section 3.3.2). In other words, the correctness and safeness of a logical pattern implementation cannot be ensured. Furthermore, implementing meta level

statements is not structure preserving; a valid implementation does not need to have the same structure as the pattern it implements. The reason is that where axioms in content patterns are imported, copied or extended, axioms in logical patterns are *instantiated*.

Design patterns are not always explicitly specified as a formal theory. They are rather *described* as step-by-step procedures larded with stereotypical examples. For instance, the procedures for specifying value partitions (Rector et al., 2004; Rector, 2005), imitating N-ary relations (Noy and Rector, 2006), classes as values (Noy, 2005), transitive propagation (Seidenberg and Rector, 2006), sequences (Drummond et al., 2006) and structured objects (Motik et al. (2007a); Hoekstra and Breuker (2008) and Chapter 7). Largely, these procedures are not intended as ontological commitment – they do not refer to entities in reality – but are rather meant to describe how certain structures can be approximated that cannot be directly expressed in OWL.

If we take definition 5.6.5 at face value, the patterns described by such procedures would only count as logical pattern when formalised as a meta theory. However, given the limitations of meta theories in OWL, and the generality of examples in pattern descriptions, the utility of a strict logical pattern does not necessarily transcend that of less formal patterns. What sets the patterns described above apart from content patterns, is that the mapping between pattern and implementation is not subject to ontological restrictions. It does not assume an ontological relation (e.g. subsumption) between the symbols in either signature, but merely ensures the transposition of the *structure* of the pattern to categories in the implementation. We can therefore introduce a third type of pattern, not described in Presutti et al. (2008), which is neither a *content pattern* nor a *logical pattern*.:

**Definition 5.6.6 (Structure Pattern)** *A* structure pattern *is an ontology design pattern for which the mapping function $M$ is* bijective*, and no semantic requirements hold as regards the relation between elements in the pattern and its implementation.*

Because of the allowed ontological disconnect between structure patterns and implementing ontologies, their implementation does not explicitly involve an ontological commitment. However, it may incorporate an *epistemological* commitment of the type identified by Bodenreider et al. (2004). Structure pattern implementations signal an intended interpretation for that part of the ontology. For instance, an OWL implementation of the N-ary relation pattern of Noy and Rector (2006) signals that even though it *is not* an N-ary relation according to the OWL semantics, it should be interpreted as such. This works in exactly the same way that the presence of OWL constructs signal the applicability of OWL semantics.

The epistemological nature of structure patterns is not a given, and can be said to decrease when their intended interpretation is more closely approximated by the standard semantics of their implementation. For example, where the intended and actual semantics of the patterns in Rector (2005); Seidenberg and Rector (2006) coincide, and those of Drummond et al. (2006); Hoekstra and Breuker (2008) are close approximations, the patterns of Noy (2005); Noy and Rector (2006) make more conservative claims.

Although it is certainly true that some of these patterns are motivated mainly by epistemological objectives, and do not have explicit ontological commitments, they do convey a certain ontological message. Design patterns may

transpose the stereotypical structure of ontological categories to new domains. In other words, the implementation of a structure pattern can still constitute a metaphor, but this depends on the strength of the ontological relation between a pattern and its implementation. Rather than complete downward taxonomic ordering as in content patterns, metaphoric use of a structure pattern depends on the reuse of *relations* between categories:

**Definition 5.6.7 (Metaphoric Use)** *The* metaphoric use *of a structure pattern $\mathcal{P}_s$ in ontology $\mathcal{O}$ requires that $\mathcal{O}$ implements $\mathcal{P}_s$, and that if $\mathcal{P}_s$ is in the axiom closure of $\mathcal{O}$ (i.e. $\mathcal{P}_s \subseteq \mathcal{O}_\cup$) then for all axioms $\alpha \in \mathcal{P}_s$ it would additionally hold that for its* properties $p_s \in \mathit{Sig}(\alpha)$ *and each corresponding property $p \in \mathit{Sig}(M(\mathit{Sig}(\alpha), \alpha))$ it holds that that $p \sqsubseteq p_s$ in $\mathcal{O}$.*

Metaphoric use of a structure pattern thus requires that the properties of an implementing ontology are at least (conceptually) subproperties of corresponding properties in the pattern. Where the implementation of content patterns requires a subsumption relation between *all* elements of the signature. To illustrate the difference, the conduit metaphor from Pinker (2007) in Section 5.6.1 cannot be implemented as content pattern, but can be a metaphoric use of a structure pattern. For instance, the former requires words to literally *be* containers for ideas, and the latter merely signals a connotation. The relational character of metaphoric use corresponds to the role of verbs in natural language metaphors (Pinker, 2007), and the prototypical representation of verbs as properties (Brachman et al., 1991).

## 5.7 Discussion

In this chapter, I discussed general methodological guidelines for ontology development, emphasised the middle-out approach for constructing an ontology and gave a characterisation of different types of ontologies available for reuse. Section 5.4.2 described the technical restrictions we need to impose to guarantee safe reuse, without compromising the ontological commitments of the reused ontology. In Section 5.5, these technical considerations are augmented with a means to make explicit part of these ontological commitments, that can be used to ensure conceptually sound extensions. The notion of *framework*, and in particular the epistemological framework, allows us to separate ontological categories from other terminological knowledge. Design patterns allow us to express partial ontological theories that can be transposed and used as metaphors.

Most, if not all of these considerations are given by the ideal of ontology sharing and reuse. Indeed, all theoretical and philosophical notions aside, someone who uses the DL semantics of OWL to build just terminological knowledge bases may not heed these requirements. Nonetheless, once a knowledge base is available on the web it will inevitably be evaluated on its merits as *reusable knowledge source*. Ontologies are not 'just' terminological knowledge bases, a view that is emphasised by the wide range of guidelines, requirements and resources outlined in this chapter. Ontology development remains an art, and no a priori guidelines exist that can ensure the quality of the result. Nonetheless, the emphasis on an ontological perspective of Section 5.5, the reuse

of existing ontologies described in Section 5.4 and the restrictions on reuse and design patterns of Section 5.6 and Section 5.4.2 are important constraints on ontology development. Although these constraints are not sufficient, they are the *necessary* conditions for quality ontologies, and perhaps give a better flavour of what an ontology *is* than any definition – let alone Gruber's – could.

The next chapter shows how the principles outlined in this chapter are applied in the construction of a common-sense based core ontology for law: the LKIF Core ontology. In Chapter 7 several important design patterns adopted in this ontology are highlighted and discussed in more detail.

# Chapter 6

# Commonsense Ontology

## 6.1 Introduction

In many ways, the corpus of legal information available today is the world wide web's little sister, at least qua structure (Hoekstra, 2008).[1] It consists of a huge volume of heterogeneous, but closely inter-linked documents. These documents are increasingly being made available in digital form as part of public accessibility projects by governments.[2] However, a major difference is that the relations between legal texts are typically expressed in natural language only. Also, these references are not always absolute, typically point to *parts* of documents, and often import an externally defined meaning of a term (de Maat et al., 2008; de Maat et al., 2006). Consolidation of such semantic references into a single representation introduces a significant maintenance issue, as legal texts are very dynamic and undergo change independently from each other. In fact, the meaning of terms in law imposes an ordering on entities in reality that can change over time, but *stays applicable* to older cases. In short, law adopts an intricate versioning scheme (Boer et al., 2004a,b; Klarman et al., 2008). The MetaLex/CEN[3] XML standard for legislative sources provides an XML schema for representing the structure and dynamics of legal texts (Boer et al., 2007c, 2002).

A semantic representation – be it for the purpose lightweight annotation, consistency checking or legal knowledge based reasoning (planning, assessment) – should take the dynamic and structural properties of legal texts into account. This is most directly reflected in the principle of *traceability*: any representation of some legal text should be traceable to its original source; it should be *isomorphic* (Bench-Capon and Coenen, 1991). A representation of some (part of) legislation is dependent on that legislation, and is therefore essentially al-

[1] This chapter is a revised version of a series of papers on the LKIF Core ontology written together with Joost Breuker and Alexander Boer (Breuker et al., 2006, 2007; Hoekstra et al., 2007, 2008).

[2] An example is the portal of the Dutch government `http://www.wetten.nl` that discloses currently active legislation

[3] CEN is the European Committee for Standardisation, See `http://www.metalex.eu`, `http://legacy.metalex.eu` and `http://www.cen.eu`

ways an *annotation* on that text. This view is central to e.g. the *structure blocks* of van Engers and Glassée (2001), where the UML/OCL representation of structural elements in legislation is organised in corresponding UML packages.[4]

The MetaLex/CEN initiative provides a standard transformation of XML encoded legal texts to RDF/XML. More elaborate, formal representations of the contents of the texts in OWL are then related to this RDF representation. A relatively uncharted application area for this approach is that of *compliance*, where the business processes of organisations (businesses and governments alike) need to be aligned with respect to some body of regulations. An additional requirement of the legal domain is that definitions of concepts should be represented in such a way that their semantic interpretation mimics the structure and applicability of the texts. This includes means to *scope* definitions with respect to particular *parts of a text*, as in e.g. deeming provisions, regarding the *temporal validity* of a text (Klarman et al., 2008), and concerning its *jurisdiction* (Boer et al., 2005b, 2007b).

One could argue that such requirements indicate the need for knowledge representation languages specific to law (as in e.g. deontic logics). However, the legal field is in one respect wholly analogous to the web in that legal information is used and incorporated in a wide variety of systems, each using the information in different ways. Also, the whole body of legal information is not maintained by a single issuer, but rather by a significant number of authorities that each publish, incorporate, extend, comply with, enforce and implement regulations. Therefore, the requirements for knowledge representation on the Semantic Web hold for representation of legal sources as well. Especially as the information exchange between those parties can benefit enormously from a well designed standard. This principle lies at the heart of the Legal Knowledge Interchange Format (LKIF) that allows for the interchange of legal knowledge between commercial vendors (Boer et al., 2007a,c).[5]

As an interchange format, one would expect LKIF to be a knowledge representation language in its own right just as e.g. the KIF of Genesereth and Fikes (1992) or its successor CommonLogic (ISO/IEC, 2007). However, LKIF rather specifies a knowledge interchange *architecture* for alternative ways of expressing different types of legal knowledge using three different representation paradigms. The first approach is characterised by the *pragmatism* of legal knowledge based system vendors. Commercial representation languages are typically rather inexpressive rule formalisms and constructed for the specific purpose of supporting features of the vendor's application suite (Gordon et al., 2007b). The second paradigm is defined by a theoretical focus on the *epistemological status* of knowledge as part of legal reasoning. In this view, a legal knowledge representation language does not purport to provide definitions, but rather characterises the use and manipulation of information to establish and *justify* legal facts. Legal knowledge is interpreted as the knowledge of reaching a legally valid conclusion. In LKIF, this strategic *control* knowledge (Clancey, 1983; Breuker and Van De Velde, 1994; van Heijst et al., 1997) is captured by a defeasible rule formalism, and more prominently by a theory of

[4]OCL: the Object Constraint Language, see `http://www.omg.org/technology/documents/formal/ocl.htm`.

[5]LKIF is developed as part of the Estrella project: European project for Standardized Transparent Representations in order to Extend Legal Accessibility (IST-2004-027655), see `http://www.estrellaproject.org`

argumentation (Boer et al., 2007a; Gordon et al., 2007a; Gordon, 2007).[6]

Where the second perspective emphasises epistemological status and legal theory – i.e. it does not include an explicit domain theory – the third approach adopts the *knowledge representation* perspective outlined in the preceding chapters. In this view, legal knowledge is characterised as that which is reasoned *with* rather than a specification of legal reasoning itself. The former can be expressed as a domain theory using the OWL DL knowledge representation formalism, while the latter is captured by meta components that implement problem solving methods over a monotonic DL knowledge base (Breuker and Van De Velde, 1994), as in the knowledge system *shells* of Marcus (1988) or Chandrasekaran and Johnson (1993). Typical legal reasoning tasks are *legal assessment*, i.e. determining whether some case or situation is allowed or disallowed given a system of normative statements, *legal planning* (Valente, 1995), but also *legal argumentation*. However, the way in which these tasks are performed is not necessarily particular to the legal domain. For instance, though legal argumentation differs from argumentation in general in that it is *adversarial* and takes place in court, the positing of arguments (hypotheses) and support (proofs, findings) is central to any type of problem solving (cf. Breuker (1994); Boer (2000), and Figure 2.9). The knowledge representation perspective allows for an isomorphic representation of the *contents* of legal sources as annotation, required for traceability and maintainability.

### 6.1.1 A Functional View

Clancey (1983) identified different types of rules involved in the medical domain theory of the MYCIN system (Section 2.3.2): identification, causal, world fact and domain fact rules. Similarly, the domain theory of *legal* knowledge based systems is not homogeneous either. Valente (1995); Valente and Breuker (1995) give a general breakdown of the types of knowledge and their dependencies involved in the legal domain. Valente's Functional Ontology of Law (FOLaw) describes the legal system as an instrument to influence society and reach certain social goals, i.e. it exists to fulfil a *function*. The legal system can be viewed as a "social device operating within society and on society, and whose main function is to regulate social behaviour" (Valente, 1995, p. 49).

Despite its name, FOLaw is not really an ontology at all, at least not in the strict sense. It describes the categories of legal knowledge we can reason about, i.e. the kinds of things one can 'know' in the legal domain and *not* the things that 'exist'. It is an *epistemological framework*, an ontology of the epistemology of law, cf. (Breuker and Hoekstra, 2004c, and Section 5.5.2). FOLaw distinguishes seven types of knowledge: commonsense knowledge, world knowledge, normative knowledge, responsibility knowledge, meta-legal knowledge, creative knowledge and reactive knowledge. Any legal knowledge based system will incorporate at least one of these knowledge types. In fact, FOLaw does not just have a functional perspective on the legal system itself, but on the different knowledge components within this system as well (Levesque, 1984, and Section 2.4.1).

The most characteristic category of legal knowledge is *normative knowledge*.

[6] A reasoner for unfolding argumentation schemes is implemented as part of the Carneades argument mapping system, `http://carneades.berlios.de/`.

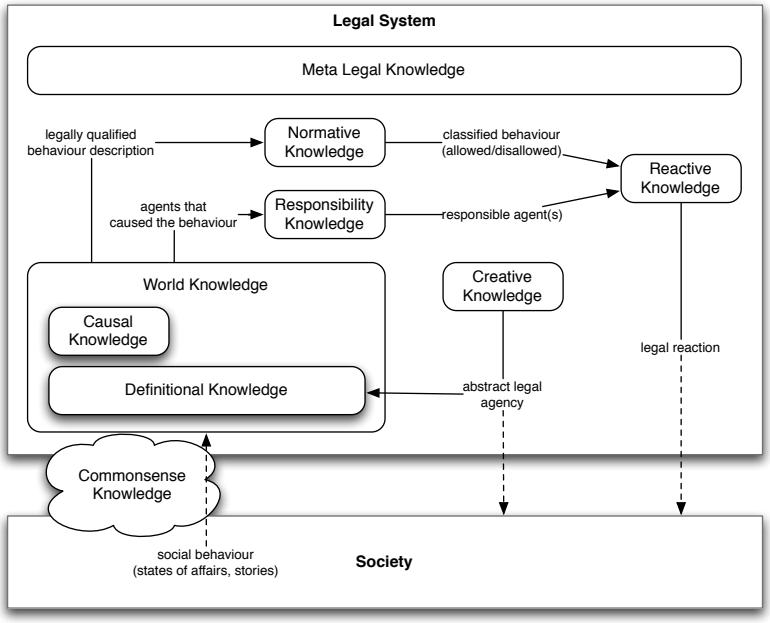


Figure 6.1: The Funtional Ontology of Law

Normative knowledge reflects the regulatory nature of law, and has two functions: prescribing behaviour, and defining a standard of comparison for social reality. A norm expresses an idealisation: what ought to be the case (or to happen) according to the will of the agent that created the norm (Kelsen, 1991). *Meta-legal knowledge* governs relations between norms, and is applied when solving normative conflicts (Elhag et al., 1999), and determining the *preference ordering* of norms (Boer, 2009). Legal principles such as *lex specialis*, *lex superior* and *lex posterior* are types of meta-legal knowledge.

Because law governs the behaviour of agents in the world, it must contain some description of this behaviour. The separation between the knowledge used to describe the world and the normative knowledge was first explicitly proposed in Breuker (1990), where this category was called *world knowledge*, or the *Legal Abstract Model* (LAM). This model functions as an epistemological interface between the legal system and social reality. World knowledge consists of *definitional knowledge* and *causal knowledge* and defines legal concepts and (causality) principles as abstractions of commonsense concepts. The knowledge needed to attribute responsibility given the legal (causal) interpretation of some story or state of affairs, is characterised as *responsibility knowledge*.

The functional ontology of law describes the path along which a formal account travels as it is under consideration of the legal system (see Figure 6.1). When an account 'enters' the legal system, it is enriched with legal vocabulary, i.e. it is interpreted and legally *qualified*. The causal relations between events described in the account are identified. This interpreted causal account is then considered for norm violations, using some body of normative knowledge (rules, legislation, precedents), and liability is determined for agents

(in)directly responsible for the violation or some harm. A reactive measure is then taken on the basis of the outcome of this process.

### 6.1.2 Purpose

The FOLaw approach is reflected in various systems, such as the TRACS system on traffic regulations of den Haan (1992, 1996), the ON-LINE legal information server of (Valente, 1995; Valente and Breuker, 1999), and the MILE system for legal assessment in the ship insurance domain developed within the CLIME project (Winkels et al., 1998, 2002).[7] In these systems, definitional and normative knowledge were most prominently defined. In particular, the world knowledge of the MILE system was represented as a domain ontology comprising 3377 concepts and 11897 relations (Boer et al., 2001). It covered about 15% of the regulations for ship classification of Bureau Veritas, a large insurance firm in France, and all of the MARPOL I and II regulations for maritime pollution.[8] In all, the ontology had 8289 references to these legal sources. The CLIME ontology was used by the MILE system for *conceptual retrieval* on the body of regulations. Queries could be formulated in terms of concepts and relations of the ontology. This set was then expanded using a finite state automaton, and references to the regulations from this result set were presented to the user. The rationale for a particular result was explained as a minimal path from the concepts in the result set to the query set. Later, the ontology was reused in a *knowledge desktop environment* that provided workflow directed document management support for insurance surveyors (Jansweijer et al., 2001).

The way in which the CLIME ontology improved the accessibility of the Bureau Veritas regulations was a typical *knowledge management* use. For instance, the *legal encoding tools* (LET) editor that was used to build the ontology offered typical management functionality, such as automatic indexing and concept extraction (Boer et al., 2001).[9] The ontology itself was relatively lightweight; the language in which it was expressed supported a fixed set of five relations and did not have formal semantics.[10] A similar perspective on ontology use is e.g. exemplified by the DALOS ontology of Agnoloni et al. (2009), the intellectual property rights ontology IPROnto (Delgado and García, 2003), and the ontology for the Iuriservice portal (Casellas et al., 2007).

However, the CLIME ontology was also the basis for incremental specification of *normative* knowledge. Concepts in the ontology were used directly in the representation of norms. This proved to be a significant bootstrapping mechanism, not just in development speed, but also as a backbone for calculating the exception structure of norms as given by the *lex specialis* principle (Winkels et al., 1999). The legal assessment engine of MILE depended on inferences over the domain knowledge in the CLIME ontology.

In a similar vein, the LKIF includes a core ontology (LKIF Core) that provides

[7] CLIME, Cooperative Legal Information Management and Explanation, Esprit Project EP25414

[8] See `http://www.bureauveritas.com` and `http://www.imo.org/Conventions/contents.asp?doc_id=678&topic_id=258`.

[9] It was in fact experience in developing the LET that initiated the MetaLex initiative of Boer et al. (2002).

[10] The five relations of CLIME were subsumption, part of, observable, measurable, related to and their inverse.

a vocabulary and a set of standard definitions of concepts common to all legal fields (Breuker et al., 2007; Hoekstra et al., 2007, 2008). In Breuker et al. (2007); Hoekstra et al. (2007, 2008) we identified three main ways for a legal core ontology to support information exchange in the setting of LKIF (cf. Valente (2005)). First of all, the ontology can serve as a resource for special, legal *inference*. Secondly, the definitions of terms in the ontology can facilitate *knowledge acquisition*, a terminological framework can facilitate the *exchange* of knowledge across multiple knowledge bases, and lastly it can be a basis for semantic *annotation* of legal information sources.

**Resource for Special, Legal Inference** Typical and abstract legal concepts are often strongly interrelated and thereby provide the basis for computing equivalencies, or paraphrases, and implications. For instance, by representing an obligation as the opposite of a prohibition, a (legal) knowledge system can make inferences that capture the notion that they are each others' inverse. A prohibition leaves all options open – except the one that is forbidden – while an obligation is unavoidable when all its requirements, or conditions, are satisfied. Although this implicit knowledge is relevant when reasoning with norms and cases, it does not express the control knowledge of reasoning (as in problem solving methods), but merely elicits the implications of declarative definitions. Specialised legal inference can be based on definitions of concepts in an ontology: an inference engine can generate the implied consequences of explicit concept definitions.

A classical example of specialised inference using the definitions in an ontology and a (general) inference engine is temporal reasoning based on Allen (1984)'s ontology of time (Section 6.3.1). To enable special inference, terms should be highly interrelated and form a coherent *cluster* with little or no external dependencies (Hayes (1985), and Section 5.2). An example of such a cluster in the legal domain is that of the terms that denote deontic qualifications. Clusters of this type are usually found at very high levels of abstraction.

**Knowledge Acquisition Support** The classical use of both top and core ontologies in knowledge representation is as a means to support knowledge acquisition (Section 5.4). If well designed and explained, they provide an initial structure to which domain terms can be attached as subclasses. Inheritance of properties and other implicit knowledge can then be used to check not only consistency, but also the extra-logical quality of the ontology: whether what is derived (classes, properties) makes sense. The use of a core or top structure that has well tested and evaluated implications, makes it easier to check whether domain refinements are not only consistent, but also arrive at inferences that correspond to what the knowledge engineer or user holds to be valid. The knowledge acquisition support of ontologies is not restricted to just ontological or even terminological knowledge. For instance, the incremental specification of normative knowledge in the CLIME ontology is an example where an ontology aids in the knowledge acquisition of non terminological knowledge.

**Preventing Loss in Translation** A legal ontology can play an important role in the translation of existing legal knowledge bases to other representation formats. In particular when these knowledge bases are converted into LKIF

as the basis for articulate knowledge serving. Similar to a translation between different natural languages, a formal, 'syntactic' translation may clash with the semantics implied by the original knowledge representation. An ontology, as representation of the semantics of terms, allows us to keep track of the use of terms across multiple knowledge bases.

**Resource for Semantic Annotation** The semantic annotation of legal sources, such as regulations and jurisprudence is an important contribution to the accessibility and maintainability of these sources. First of all, an ontology can be a source for information retrieval, such as e.g. the CLIME ontology. Secondly, the status of an officially sanctioned legal text is primarily determined by its relation to other legal texts (Boer et al., 2004a). This status can be made explicit by expressing it using ontologically defined relations. In fact, these relations do not just hold between the texts themselves, but between the formal representations of their content as well (Klarman et al., 2008).

Besides the general requirements for knowledge representation ontologies outlined in the preceding chapters, the ontology is to contain a core set of definitions for describing specific legal terms. We have seen that law can be viewed as an instrument used by the legal and political system to identify and control situations and events in social interaction (Valente, 1995). By far the bulk of social situations, be it in our family life, at work, related to transport, property, crime, etc. is not described in specialised technical terms: their meaning is part of common sense. For instance, the conflicts and problems brought to court – legal cases – are initially described using common sense terms, and are gradually translated into legal technical terminology in the process of coming to a decision. For this legal qualification to be possible, the gap between the vocabulary of a case and legal terminology needs to be bridged (Winkels and de Bruijn, 1996). The possibility of legal qualification in general – by legal professionals – is a strong indication that the vocabularies of legal knowledge and common sense are not disjoint. Legal terminology can be reduced to the actual societal events and states governed by law. In other words, the basic categories of the LKIF ontology should reflect Valente's view that legal world knowledge is an abstraction of common sense.

A third requirement for LKIF is given by the ideal of the Semantic Web to achieve understanding both between web services and between human users (Chapter 3). In fact, a commonsense perspective is also applicable to any serious endeavour towards a Semantic Web. As the the web is about the most diverse information source we know today, a common sense oriented ontology would certainly be an important first step to more uniform web-based knowledge representation. Conversely, as hinted at in the introduction, the distribution of legal information across various strongly interconnected sources is a demanding use case for semantic web technology (Hoekstra, 2008). An important requirement is therefore that the LKIF core ontology should be represented using the DL profile of OWL.

The following sections describe the theoretical and methodological framework against which the LKIF core ontology has been developed. Section 6.2.1 discusses the perspective used in its construction in relation to five other ontologies. The methodology used to construct the LKIF ontology is discussed in

Section 5.2. Section 6.3 introduces the modules and most important concepts of LKIF Core.

## 6.2 Developing a Core Ontology

Given a commonsense perspective, it is expected that (at least parts of) existing ontologies would be reusable. Either as a source of inspiration or for the purpose of full formal reuse of definitions. This would hold in particular for top ontologies that include legal terms, as for instance listed in Casanovas et al. (2006). Unfortunately, it turned out that the amount of reuse and inspiration was rather limited. Not only do existing ontologies diverge on the approach, coverage and knowledge representation language used; those ontologies that do claim a common sense or similar perspective differ in their conception as to what such a perspective *means*.

### 6.2.1 Ontology Integration

This section evaluates several of the foundational and core ontologies introduced in Section 5.4.1 with respect to a potential contribution to LKIF Core. The main requirement is their suitability to enable the primary roles of the LKIF ontology outlined in the previous section. We pay specific attention to their definition of commonsense and legal terms, and possibilities for safe reuse (Section 5.4.2).

#### Suggested Upper Merged Ontology

The SUMO ontology of Niles and Pease (2001) brings together insights from engineering, philosophy, and information science. It provides definitions for general purpose terms, is intended as a unifying framework for more specific domain level ontologies. As a starting point for a legal core ontology SUMO has several drawbacks. First of all, it does not readily provide definitions of terms relevant to the legal field – e.g. its coverage of mental and social entities is limited. Because of the way in which SUMO is constructed, it has a bias towards more abstract and theoretical insights coming from engineering and philosophy. Although it is non-revisionist, as in e.g. the distinction between objects and processes, it does not have a real commonsense basis. Furthermore, as discussed in Section 5.4.1 SUMO is a *foundational ontology* and uses meta modelling, such as in the definitions of classes, binary relations and sets. As SUMO is represented in the expressive language KIF, and more recently CommonLogic, this practice is not fundamentally problematic. However, it means that safe reuse of SUMO definitions in the construction of an OWL DL ontology is not possible.

#### Descriptive Ontology for Linguistic and Cognitive Engineering

DOLCE is part of the WonderWeb library of foundational ontologies, cf. Masolo et al. (2003); Gangemi et al. (2002). It was meant as a reference point for the other ontologies in the library, to make explicit the differing rationale and alternatives underlying the ontological choices of each ontology. Rather than a

coherent set of ontological commitments, it captures a range of alternatives. This way, the library would form a network of different but systematically related ontology modules. The relation between an ontology, available in the library, and the DOLCE ontology expresses its ontological commitment to particular ontological options. DOLCE was therefore never presented as the foundational ontology it is currently regarded as, but it has been successfully used as such in a large number of projects.

DOLCE is very much an ontology in the philosophical tradition, and differs from the knowledge representation perspective in two significant ways. Firstly, its perspective is philosophical with respect to its *content*, i.e. it is aimed to directly represent reality. And secondly, it is subject to the epistemological promiscuity of philosophical ontology because it is rather an extension of the knowledge representation formalism at the *ontological level* (Guarino, 1994), than a model expressed using that formalism. The meta-level character of DOLCE means that the ontology is not a representation of knowledge, but of the terms used to describe knowledge – in the same way that the constructs of OWL are. Like SUMO, DOLCE was originally specified in first order logic and the highly expressive KIF language. Its OWL DL representation (DOLCE-Lite) is more restrictive, e.g. it does not consider temporal indexing and relation composition.

The DOLCE ontology is *descriptive*, and is based on the stance that "the *surface structure* of natural language and human cognition"[11] is ontologically relevant. It is argued that this perspective results in an ontology that captures cognitive artefacts more or less depending on human perception, cultural imprints and social conventions, and *not* deep philosophical insights of Ontology. DOLCE thus claims an approach that fits more with a commonsense perspective than the science perspective of SUMO. However, the suggestion that this surface structure has any bearing on common sense is not based on evidence. Rather, the methodological commitment to the *surface* structure of language and cognition almost inevitably resulted in an intricate framework of theoretical notions needed to encompass the idiosyncrasies of human language use. Alternatively, rather than constructing an ontology by studying reality through the kinds of categories implicit in natural language, a pragmatic approach based more directly on the conceptualisation of reality we use and live by in our daily routine is more appropriate (Breuker and Hoekstra, 2004a; Hoekstra et al., 2008, and Section 6.2.2).

#### Core Legal Ontology

Over the years, DOLCE has been extended in several ways. DOLCE+ is the extension of DOLCE with a theory on descriptions and situations (also called D&S, Gangemi and Mika (2003)). CLO, the Core Legal Ontology (Gangemi et al., 2005) extends DOLCE+ even further and defines legal concepts and relations based on its formal properties. CLO was designed to support both the definition of domain ontologies, a juridical Wordnet, and the design of legal decision support systems. To a large extent these goals correspond with the requirements of the LKIF ontology.

CLO conceives the legal world as a *description* of social reality, an ideal view of the behaviour of a social group. It builds on the D&S distinction between

[11] Emphasis by the authors, Masolo et al. (2003)

*descriptions*, and *situations*. Examples of legal descriptions, or *conceptualisations*, are the *contents* of laws, norms, and crime types. These descriptions constrain legal situations, i.e. legal *facts* of *cases*, such as legal, relevant non-legal and juridical states of affairs. Every legal description *classifies* a state of affairs. More precisely, a legal description is the reification of a theory that formalises the content of a norm, or a bundle of norms. A legal case is the reification of a state of affairs that is a logical model of that theory. A description is satisfied by a situation when at least some entity in the situation is classified by at least some concept in the description. Classification in CLO is thus not DL classification, and it is unclear as to what extent the two interpretations are compatible.

As described earlier, the legal system as description, or rather *pre*scription, of reality is not new, cf. Valente (1995, others). However, the CLO distinction between descriptions and situations is rather one dimensional in that it does not commit to an ontological view of the *kinds* of descriptions involved. In line with the DOLCE ontology, it confounds the distinction between representation and the *represented* with representation and *reality*. Although it introduces new levels of abstraction by reification, it does not provide ontological categories that can be used to describe the knowledge at these levels. In a language that itself can be conceived as providing the means to construct descriptions of reality (situations), such as OWL DL, it is unclear what the epistemological status of the classes 'description' and 'situation' *themselves* is. For instance, what is the difference between an individual description being classified-in-the-OWL-sense as some description class, and a situation class being classified-in-the-CLO-sense by that same description?

As CLO relies on a subset of DOLCE for the definition of *elements* of situations, it is subject to the same criticism with respect to its commonsense perspective. Moreover, the lack of ontological commitment at the level of descriptions undermines its suitability for knowledge acquisition support in a legal setting as well. Although for sure a norm can be described as some description of a situation, it is not the norm-as-description that uniquely characterises what a norm *is*. This holds especially for less obvious 'descriptions' (in CLO terms), as e.g. damage or right of way.

### CYC

CYC is a huge unified ontology of commonsense concepts (Lenat et al., 1990; Lenat, 1995). Although the project has started as early as 1984, its general set-up corresponds to that of later large scale ontology efforts. The main motivation for the Cyc Project was that all software programs would benefit from the availability of commonsense knowledge. This idea is not new, and was acknowledged in the early years of AI: "A program has common sense if it automatically deduces for itself a sufficiently wide class of immediate consequences of anything it is told and what it already knows" (McCarthy, 1959, p.2).

The idea is that, when enough commonsense knowledge is represented, and a certain threshold is reached, a quantum-leap ("The Singularity")[12] would enable CYC to expand its knowledge through guided learning (as a human child would). This theory is in line with Minsky's ideas about how computers can become intelligent beings: add enormous amounts of commonsense know-

[12]Indeed, CYC is a much-hyped project, and has received a lot of criticism because of it.

ledge (Minsky, 1982, 1984). This basic knowledge about the workings of the world would finally allow you to send the kid to school. With currently over 300K concepts, the knowledge base seems well under way in reaching this threshold, however we still have to see the first results.[13]

The upper part of the CYC ontology is claimed to express a commonsense view and indeed it is more concrete than either SUMO or DOLCE. On the other hand, from a methodological point of view, the CYC approach is not very satisfactory either. Technically, CYC qualifies rather as a terminological knowledge base than as an ontology proper.[14] Instead of a meticulous study of the actual workings of the world, as in SUMO, or the surface structure of language and cognition, as in DOLCE, it seems the approach followed is to have a bunch of knowledge engineers simply put everything they know into the CYCL formalism.[15] This procedure results in a large portion of the knowledge base being decidedly non-ontological, but rather context dependent *frameworks* (see Section 5.5.2).

Furthermore, CYC suffers from two more technical impediments for reuse. Firstly, like SUMO and DOLCE, it is specified in the very expressive CYCL representation language, which is based on first-order predicate calculus. Recently a port of the publicly available OpenCYC effort has been made available in OWL Full, but again, this does not cover the full semantics of the ontology.[16] CYCL admits meta modelling, which indeed is used liberally throughout the ontology – even more so than in DOLCE and SUMO. Secondly, the sheer size of the knowledge base – as with most unified ontologies – introduces significant reasoning overhead for even the simplest tasks. As such, CYC seems more suitable for direct inclusion in a knowledge based system than as a conceptual coat rack for ontology development on the Semantic Web.

### 6.2.2 Ontology Reuse

Thus far, the ontologies we reviewed do not appear to meet the requirements for the top structure of a legal core ontology. Although in the past few years the ontologies underwent changes and extensions, these results are in line with the outcome of an earlier review (Breuker and Hoekstra, 2004a). Firstly, the ontologies are specified at multiple (meta) levels of abstraction, using very expressive languages, which limits possibilities for safe reuse.

Furthermore, in all three ontologies those concepts that are of relevance to law are either scarce and under specified, or overly theoretical. In particular, our requirement that a legal core ontology should be built on a commonsense conception of reality is not met. Where a commonsense perspective is claimed, it is not motivated, explained or substantiated. The common sense of CYC is in fact *common knowledge*, or rather "human consensus reality knowledge" (Lenat et al., 1990, p. 33), i.e. that knowledge of things most humans will concede to exist in reality. In contrast, the DOLCE and SUMO ontologies do not commit

[13] The online game FACTory for teaching CYC is online at `http://game.cyc.com`.

[14] A random concept search just returned 'Laurel Goodwin', the acclaimed American actress who played the memorable role of Yeoman J.M. Colt in the 1966 pilot of Star Trek. See `http://sw.cyc.com/2006/07/27/cyc/LaurelGoodwin`.

[15] See `http://www.cyc.com/cycdoc/ref/cycl-syntax.html`

[16] See `http://www.opencyc.org` for an online browser and `http://www.cyc.com/2004/06/04/cyc` for the OWL Full version

to the ontological relevance of human consensus, but rather aim to provide an ontological grounding for *all* knowledge. They do this in two distinct ways. SUMO is based on scientific knowledge of objective reality, and for DOLCE the way in which we apply and consciously report our knowledge is ontologically relevant.

Both approaches are generic enough to be the basis for an ontology that describes our common sense, in the sense of common knowledge. However, in our view, common sense refers not to some common body of knowledge of reality, but rather to the commonality of the scope and detail of that knowledge as it manifests itself in individual persons: it is a *level of description*, much akin to the *basic level* of Lakoff (1987) (see Section 5.2). Characteristic of this level is not just the *kind* of things we commonly know, but more importantly, the way in which this knowledge is *structured*. But for the last requirement, a philosophical approach would be perfectly adequate. However, a commonsense ontology should not be specified in highly specific, theoretical jargon, but should rather have a commonsense structure of its own.

#### LRI Core

The review in Breuker and Hoekstra (2004a) motivated a decision to develop a legal core ontology to support the development of ontologies for criminal law in various European countries, as part of the e-Court project.[17] This ontology, *LRI Core*, was developed with a set of design goals similar to that of LKIF Core, cf. Breuker and Hoekstra (2004c,a). What sets LRI Core apart from other ontology efforts is that its definitions were aimed to be verifiable through empirical research on how humans relate concepts in actual understanding of the world; and not about revisionist views of how we should view the world as it actually *is* (as e.g. in correct theories of the physical world) or as it makes up a parsimonious view on reality (as e.g. in philosophical views on the main categories of description).[18] This kind of empirical evidence can range from cluster analysis of semantic distance between terms, to neuro-psychological evidence.

Central to this effort is the view that common sense is rooted in a conceptualisation that is at its heart the result of our evolution. This conceptualisation is developed in order to deal with a dynamic and potentially dangerous environment. Our capacity to move, perceive and interact with reality has led to increasingly complex cognitive representations. These representations range from hard-wired abstraction in our perception system, such as the ability to perceive straight lines and angles at a mere 2 neurones away from the retina, via the inborn syntheses of perceptual input into basic properties, to – eventually – the representations accessible to conscious thought.

This range of increasingly abstract and complex representations of reality defines an axis that indexes organisms in successive stages of evolutionary development, e.g. from viruses and bacteria to multi-cell organisms, insects, mammals, primates and finally homo sapiens. In other words, the compet-

[17] Electronic Court: Judicial IT-based Management. e-Court was a European 5th Framework project, IST 2000-28199.

[18] For instance the classical distinction between abstract and concrete concepts in philosophy does not fit well with human understanding of the dynamics of the world in terms of physical causation, intention and life. The commonsense explanation of an event may involve all three categories.

encies of our genetic ancestors give insight in what the roots of our common sense are. These roots may well be hidden too deep to be accessible to conscious thought and introspection. Nonetheless, on the basis of insights in cognitive science we can make several basic distinctions that go well beyond a mere hunch. As most are of relevance to the LKIF Core ontology as well, they are briefly outlined below.

**A Cognitive Science Perspective** Given that the physical environment is relatively stable – the notable exceptions being day-night cycles, changing weather conditions and the occasional geological perturbation – the perceptual apparatus of most organisms is tuned to register the slightest change occurring against this stable canvas. Of particular relevance is the ability to be aware of changes induced by other organisms. Firstly, the presence of another organism may present an opportunity for *reproduction* and sustaining *metabolism* (i.e. by eating). And secondly, their presence may signify a direct threat to an organisms' existence. The result is a prominent distinction in cognition between *'background'* and *'foreground'* (Hobbs, 1995). In general our awareness is directed to the *dis*continuity of change rather than spatial arrangements of objects or historical continuity. The ability to perceive stability is enabled by episodic memory, though it requires some conception of the physical constraints underlying this stability. An example is the general rule that physical objects keep their position unless subjected to a change in the exertion of force. Changes occur against the canvas of temporal and spatial positions, and the speed of a change is indicative of whether it becomes foreground or remains in the background.

In LRI Core, the view that knowledge serves to interpret occurrences in the world was reflected by a contrast between concepts on the one hand, and individuals and their occurrence (instance) on the other. Furthermore, the cognitively primary distinction between static and dynamic elements in the (physical) world is reflected by differentiating objects and processes. Objects have extensions in space where processes have extensions in time, but are contained by objects. Processes reflect a causal explanation of change. The notions of space and time do not just play a role as the extension of objects and processes but can indicate positions as spatio-temporal referents. The spatio-temporal position of of objects is not inherent: a change in position does not constitute a change *in* an object.

Very recent – at least in evolutionary terms – mammals developed the ability to attribute intentions to other animals. Because of the intentional interpretation of behaviour, change is no longer private to physical causation but to the *actions* of other *agents* as well. Actions are always intentional "under some description" (Davidson, 2001, ch.3): performing an action comes down to the initiation of processes that bring about some intended change or state.[19] Paradoxically enough, taking into account the mental state of other animals precedes the ability to consciously reflect on our *own* mind. And although this was long thought to be a skill exclusive to humans, it has been shown that our next of kin – chimpanzees, bonobo's – have self awareness as well.

[19] Note that this allows the same events to be not intentional under some other description. However, if we take the events to constitute an *action*, this is a description that presupposes intentionality.

Because mental representation of other mental representations is a fairly recent accomplishment, the models we construct reuse many parts of the conceptualisation originally developed to deal with physical reality. We think and speak of mental *processes* [20] and mental *objects* in the same terms we use to talk about the physical world. In other words, these terms are metaphors of similar physical notions. Some, i.e. Lakoff and Núñez (2000), even argue that the same mechanism is used to construct the highly abstract notions of mathematics.

Social awareness and self awareness are the most important prerequisites for complex social behaviour. First of all, they enhance the predictability of our environment by allowing us to take the possible intentions of other agents into consideration for planning and control. Secondly, it allows us to share plans with other agents. Repeated execution of such co-operative plans can render them institutionalised by a community. Plans make extensive use of *roles* to specify expected or default behaviour. The ability to play a certain role expresses a (recognised) competence, sometimes acknowledged as a social 'position'. In LRI Core, roles played a central part in the construction of social structures.

The LRI Core ontology distinguished four 'worlds':

1. A *physical world*, divided by processes and objects, each containing matter and energy.

2. A *mental world*, containing mental objects and mental processes. The mental world is connected to the physical world through actions, which translate an intention into some physical change.

3. A *social world*, built from mental objects such as roles

4. An *abstract world*, which contains only formal, mathematical objects.

Because of its grounding in cognitive science and its explicit common sense perspective, the characteristics of the LRI Core ontology are relatively similar to those intended for the LKIF ontology – especially in comparison to the other ontologies we discussed so far. And it is indeed true that it can in many ways be seen as the direct precursor of the LKIF ontology.[21]

Nonetheless, the LRI Core ontology falls short in several important respects. Though it is specified in OWL DL, most concepts in the ontology are under specified. They are defined by subsumption relations, and are only sparsely characterised using more complex class restriction. Furthermore, the distinction between the different worlds in LRI Core is fairly absolute, and no theory is provided as to how they are connected and interact. Thirdly, apart from a relatively well-developed characterisation of roles, LRI Core is relatively underdeveloped with respect to the mental world. For instance, it emphasises physical objects, processes, energy and substance while remaining rather sketchy with respect to their mental counterparts. In part the limitations of LRI Core can be ascribed to an unprincipled top down methodology (see Section 5.3).

Concluding, although the perspective and main distinctions of LRI Core were used as inspiration in the construction of the LKIF ontology, it is not

[20] Also: mental actions, e.g. in (trying) to control one's thoughts.

[21] This is not really surprising as there exists an overlap between the developers of LRI Core and LKIF Core.

| # | Importance | Abstractness | Legal Relevance |
|---|---|---|---|
| 1 | Law | Deontic operator | Civil law |
| 2 | Right | Law | Law |
| 3 | Jurisdiction | Norm | Legal consequence |
| 4 | Permission | Obligative Right | Legislation |
| 5 | Prohibition | Permissive Right | Obligation |
| 6 | Rule | Power | Right |
| 7 | Sanction | Right | Authority |
| 8 | Violation | Rule | Deontic operator |
| 9 | Power | Time | Duty |
| 10 | Duty | Anancastic Rule | Jurisdiction |
| 11 | Legal Position | Existential Initiation | Legal Fact |
| 12 | Norm | Existential Termination | Legal Person |
| 13 | Obligation | Potestative Right | Legal Position |
| 14 | Permissive Right | Productive Char. | Legal Procedure |
| 15 | Argument | Absolute Obl. Right | Liability |

Table 6.1: Fifteen highest scoring terms for *importance*, *abstractness*, and *legal relevance*.

simply a specialisation of LRI Core. Not only is it built from the ground up, the methodology by which it is constructed forces a broader, more concrete, and more rigorous definition of concepts and relations. First, the scope of the ontology was determined by selecting a core set of basic concepts. These concepts were organised in modules, and formed the basis for a middle-out construction of the ontology.

### 6.2.3 Scope

The LKIF Core ontology is to provide a core set of definitions to be used across the legal domain. Not only should these definitions be relevant to law, they should exist at a level that allows for a more specific interpretation of terms in sub domains of the field such as criminal or private law. Furthermore, a focus on overly theoretical and abstract concepts impedes the connection to common sense. In other words, we are looking for the *basic concepts* of law (Lakoff, 1987, and Section 5.3.1) that allow us to construct the ontology in a middle-out fashion (Uschold and King, 1995, and Section 5.3).

As discussed in Section 5.3.1, what concepts count as basic depends on the expertise present in a community of practice. Where 'manslaughter' may be basic to legal professionals, the average citizen is unlikely to be able to explain how it is different from murder. To obtain an appropriate level of description, the basic concepts in the ontology should reflect the heterogeneity of its prospective users. In Breuker et al. (2007) we identified three main groups of users: *citizens*, *legal professionals* and *legal scholars*. Although legal professionals use the legal vocabulary in a far more precise and careful way than laymen, for most of these terms the extent of common understanding is sufficient to treat their usage as analogous (Lame, 2004). Nonetheless, a number of basic terms, such as 'liability' and 'legal fact', have a specific technical legal meaning.

Several representatives from every partner in the Estrella consortium, covering each of the three user groups were asked to supply their Top 20 of legal concepts. These lists were combined with terms frequently occurring in literature such as jurisprudence and legal text-books. The resulting list comprised a

| # | Legal vs. Commonsense | Common vs. Specific |
|---|---|---|
| 1 | Perpetual injunction | Absolute right |
| 2 | Cantonal Court | Fact |
| 3 | Elements of claim | Jurisdiction |
| 4 | Provisional rule | Law |
| 5 | Solicitor | Lawyer |
| 6 | Interlocutory injunction | Obligation |
| 7 | Other directed norm | Permission |
| ... | ... | ... |
| 242 | Critical question | Narcotics |
| 243 | Dialogue | Pensioner |
| 244 | Dialogue type | Veteran |
| 245 | Family | Deformation |
| 246 | Criteria | Genocide |
| 247 | Critical discussion | Enthyememes |
| 248 | Mafia | Mafia |

Table 6.2: Top and bottom seven terms for *legal vs. common sense* and *common vs. specific legal term*

grand total of about 250 terms. Users were asked to assess each term from this list on five scales:

- the perceived level of *abstraction*,
- the perceived *relevance* for the legal domain,
- the degree to which a term is *legal* rather than *common-sense*,
- the degree to which a term is a *common legal term* (as opposed to a term that is specific for some sub-domain of law),
- and the degree to which the expert thinks this term should be *included* in the ontology.

The resulting scores were used to select an initial set of 50 terms plus several reused from other ontologies (see Tables 6.1 and 6.2), and formed the basis for the identification of clusters and the development of the LKIF Core ontology. In the end, several technical terms were included in the ontology because they capture an abstract meaning central to law (Sartor, 2006). Furthermore, such terms provide structure to the relation between more generally understood legal terms.

## 6.3 Ontology Modules

The preceding sections introduce the requirements for LKIF Core as a knowledge representation ontology for the legal domain. It is based on insights from cognitive science and uses the well established middle-out methodology. With these considerations in mind, the LKIF ontology was initially designed as a collection of eight ontology modules: *expression*, *norm*, *process*, *action*, *role*, *place*, *time* and *mereology*, cf. Breuker et al. (2006). This collection was later extended with a top ontology and two more ontology modules (*legal_action*, *legal_role*), see Figure 6.2 (Hoekstra et al., 2007; Breuker et al., 2007).[22] Each of these mod-

[22] Version 1.0 of the ontology also included two frameworks (*time_modification* and *rules*) but these are not discussed here.

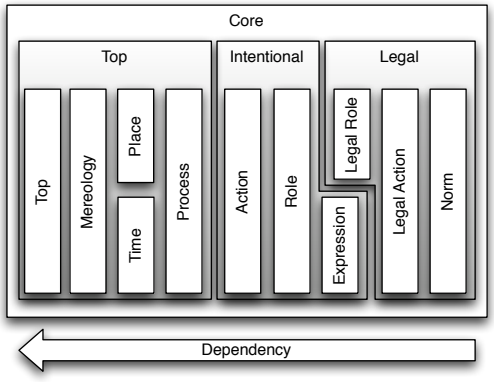


Figure 6.2: Dependencies between LKIF Core modules.

ules contains a relatively independent cluster of concepts that was represented using OWL DL in a middle-out fashion: for each cluster the most central concepts were represented first.[23]

The ontology modules are organised in three layers: the *top* level (Section 6.3.1), the *intentional* level (Section 6.3.2) and the *legal* level (Section 6.3.3). These layers correspond to the different *stances* one can adopt towards a domain of discourse, and are inspired by the work of Dennett (1987) and Searle (1995). Dennett identified three stances we can adopt for explaining phenomena in the world: the *physical* stance, used for explaining behaviour in terms of of the laws of physics, the *design* stance, which assumes a system will behave according to its design, and the *intentional* stance, which can be adopted to explain the behaviour of rational agents, with beliefs, desires and intentions. The first two correspond roughly to the top level of LKIF Core, where the intentional stance is captured by the intentional level:

> "that feature of representations by which they are *about* something or *directed at* something. Beliefs and desires are intentional in this sense because to have a belief or desire we have to believe that such and such is the case or desire that such and such be the case. Intentionality, so defined, has no special connection with intending. Intending, for example, to go to the movies is just one kind of intentionality among others."
>
> (Searle, 1995, p.7)

The LKIF ontology thus adds a *legal* layer, containing concepts that are only sensible from a legal perspective. Accordingly, we represent social and legal concepts as social constructs separate from the physical entities having such imposition, e.g. persons are said to play roles and objects fulfil functions (Searle, 1995).

This distinction does not hold for the layers as a whole: they should not be confused with stratified meta levels. Each layer introduces a straightforward

[23]The ontology was developed using TopBraid Composer (`http://www.topbraidcomposer.com`) and Protégé 4.0a (`http://protege.stanford.edu`).

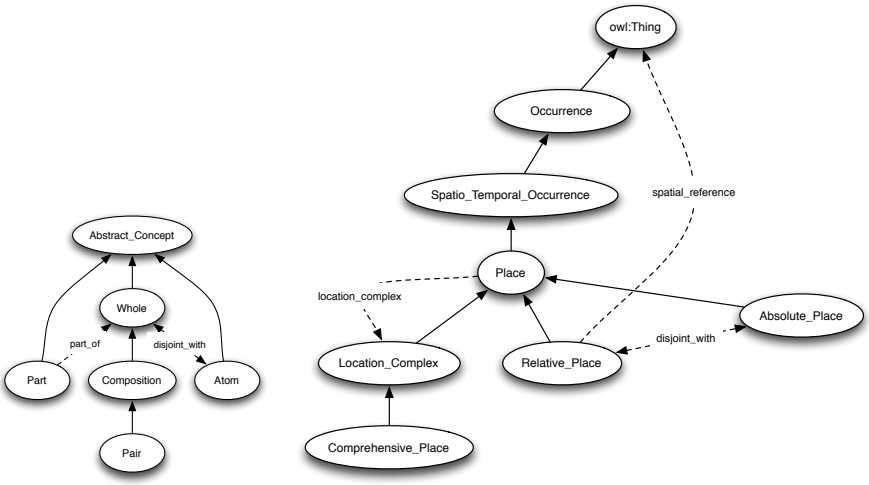


Figure 6.3: Place and Mereology related concepts.

extension of existing definitions. At each level, concepts are expressed in terms of concepts defined at a higher level of abstraction, adding new organising structures (such as properties) where necessary. However, module extension does not follow the requirement of safe reuse (Section 5.4.2. As regards its ontological commitment the ontology should be regarded as *a whole*. This means that safety of reuse of the ontology by third parties can only be assessed with respect to the combination of all ontology modules. This methodology ensures a modular set-up that improves reusability and allows extensions to commit to the ontology at one of the levels, without compromising compatibility with extensions that commit to a different level.

The following sections give a concise overview of the modules of LKIF Core and their principal concepts. They do not provide exact OWL DL definitions of these concepts but rather focuses on the necessary ingredients for a legal core ontology. Several principal concept definitions are discussed in detail in the next chapter.

### 6.3.1 First Things First: The top-level

The description of any legally relevant fact, event or situation requires a basic conceptualisation of the context in which these occur: the backdrop, or canvas, that is the physical world. Fundamental notions such as location, time, mereology and change are indispensable in a description of even the simplest legal account. The top level clusters of the ontology provide (primitive) definitions of these notions, which are consequently used to define more intentional and legal concepts in other modules. The most general categories of the LKIF ontology are based on the distinction between 'worlds' of LRI Core. We distinguish between mental, physical and abstract entities, and occurrences. Mental entities reside in the human mind, and only have a temporal extension. Physical entities exist independent of human experience, and have a spatial extension as well. Although these categories are superimposed on the concepts in the top

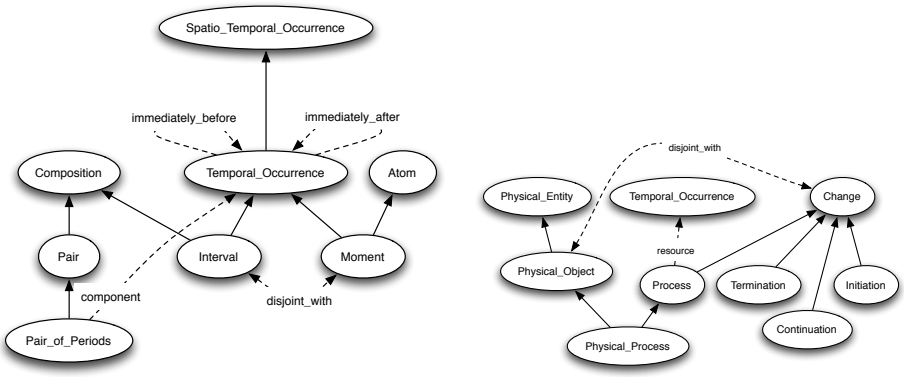


Figure 6.4: Concepts related to time and change.

level clusters, they were not directive in in their design.

Mereological relations allow us to define parts and wholes; they can be used to express a systems-oriented view on concepts. Examples are functional decompositions, and containment characteristic for many frameworks (Figure 6.3). Mereology lies at the basis of definitions for *places* and moments and intervals in *time*. The ontology for places is based on the work of Donnelly (2005) and adopts the Newtonian distinction between *relative* and *absolute* places. A relative place is defined by reference to some thing; absolute places are part of absolute space and have fixed spatial relations with other absolute places. A Location_Complex is a set of places that share a common reference location, e.g. the locations of all furniture in a room. See figure 6.3 for an overview of concepts defined in the place module. Of the properties defined in this module, meet is the most basic as it is used to define many other properties such as abut, cover, coincide etc. See Breuker et al. (2007); Donnelly (2005) for a more in depth discussion of these and other relations. Currently this module does not define classes and properties that express direction and orientation.

Similar relations can be used to capture notions of time and duration. We adopt the theory of time of Allen (1984); Allen and Ferguson (1994), and distinguish between the basic concepts of Interval and Moment. Intervals have an extent (duration) and can contain other intervals and moments. Moments are points in time, they are atomic and do not have a duration or contain other temporal occurrences (see figure 6.4). Relations between these temporal occurrences can be used to express a timeline. Allen introduced the meet relation to define two immediately adjacent temporal occurrences. To discern between the temporal meets relation and its spatial counterpart (Donnelly, 2005), this relation is called immediately_before. Where the spatial relation is unrestricted with respect to direction, the temporal meet relation is directed and asymmetric. It is used to define other temporal relations such as before, after, during. Locations and temporal entities are used to define the extension of mental and physical entities, e.g. the time when you had a thought, the location where you parked your car. They are *occurrences* and do not have extensions themselves, they *are* extensions.

With these classes and properties in hand we introduce concepts of (invol-

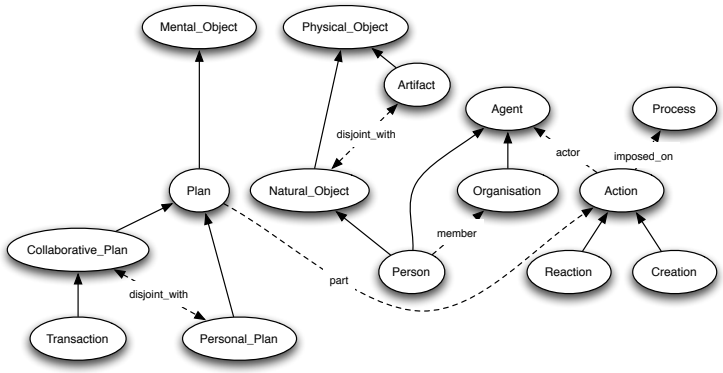


Figure 6.5: Actions, agents and organisations.

untary) change. The process ontology relies on descriptions of time and place for the representation of duration and location of changes. A Change is defined as a difference between the situation before and after the change. It can be a functionally coherent aggregate of one or more other changes. More specifically, we distinguish between Initiation, Continuation and Termination changes.

Changes that occur according to a certain recipe or procedure, i.e. changes that follow from causal necessity are Processes. They thus introduce causal propagation and are said to *explain* the occurrence of change. Processes in LKIF Core are similar to the fluents of event calculus (van Lambalgen and Hamm, 2005). However, the ontology does not commit to a particular theory of causation and we consider the perspective generic enough to enable the definition of various disparate conceptions of causal relata. Contrary to changes, processes are both spatially and temporally restricted. They extend through time – they have duration – and are located at some temporal and spatial position. We furthermore distinguish Physical_Processes which operate on Physical_Objects.

### 6.3.2 The Intentional Level

Legal reasoning is based on a common sense understanding that allows the prediction and explanation of intelligent behaviour. After all, it is only the behaviour of rational agents that is governed by law. The modules at the intentional level include concepts and relations necessary for describing this behaviour: Actions undertaken by Agents in a particular Role. Furthermore, it introduces concepts for describing the mental state of these agents, *subjective entities* such as their Intention or Belief, but also communication between agents by means of Expressions (Searle, 1995, and Section 7.3.2).

The class of agents is defined as the set of things which have intention and can be the actor of an action: they may perform the action and are potentially liable for any effects caused by the action (see Figure 6.5). An action is 'intentional under some description' (Davidson, 2001), they constitute an intentional interpretation of a process that reflects the changes brought about by some agent in realising his intentions. Agents are the medium of an action's intended outcome: actions are always intentional. The intention held by the agent usu-

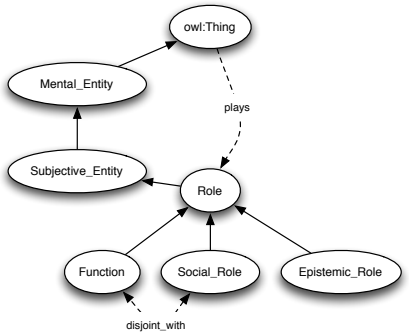


Figure 6.6: Roles.

ally bears with it some expectation that the intended outcome will be brought about: the agent believes in this expectation. The actions an agent is expected or allowed to perform are constrained by the *competence* of the agent, sometimes expressed as *roles* assigned to the agent. Because actions coincide with processes, they can play a role in causal propagation, allowing us to reason backwards from effect to agent (see Section 7.5). Actions can be creative in that they initiate the coming into existence of some thing, or the converse terminate its existence. Also, actions are often a direct *re*action to some other action.

The LKIF Core ontology distinguishes between Persons, individual agents such as "Joost Breuker" and "Pope Benedict XVI", and Organisations, aggregates of other organisations or persons which acts 'as one', such as the "Dutch Government" and the "Sceptics Society". Artefacts are physical objects designed for a specific purpose, i.e. to perform some Function as instrument in a specific set of actions such as "Hammer". Persons are physical objects as well, but are not designed (though some might hold the contrary) and are subsumed under the class of Natural_Objects. Note that natural objects can function as tools or weapons as well, but are not *designed* for that specific purpose.

The notion of roles played an important part in recent discussions on ontology (Steimann, 2000; Guarino and Welty, 2002; Masolo et al., 2004; Loebe, 2007). A Role is a Subjective_Entity that specifies standard or required properties and behaviour of the entities playing the role (see Figure 6.6). Roles not only allow us to categorise objects according to their prototypical use and behaviour, they also provide the means for categorising the behaviour of other agents. They are a necessary part of making sense of the social world and allow for describing social organisation, prescribe behaviour of an agent within a particular context, and recognise deviations from 'correct' or normal behaviour. Indeed, roles and actions are closely related concepts: a role defines some set of actions that can be performed by an agent, but is conversely defined by those actions. The role module captures the roles and functions that can be played and held by agents and artefacts respectively, and focuses on *social* roles, rather than traditional thematic or relational roles.

A consequence of the prescriptive nature of roles is that agents connect expectations of behaviour to other agents: intentions and expectations can be

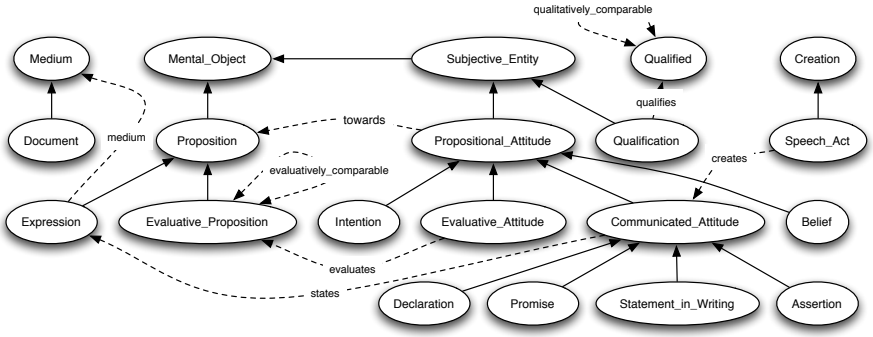


Figure 6.7: Propositions, Attitudes and Expressions.

used as a model for intelligent decision making and planning.[24] It is important to note that there is an *internalist* and an *externalist* way to use intentions and expectations. The external observer can only ascribe intentions and expectations to an agent based on his observed actions. The external observer will make assumptions about what is *normal*, or apply a *normative* standard for explaining the actions of the agent (Boer et al., 2005a; Boer, 2009).

The expression module covers a number of representational primitives necessary for dealing with Propositional_Attitudes (Dahllöf, 1995). Many concepts and processes in legal reasoning and argumentation can only be explained in terms of propositional attitudes: a relational mental state connecting a person to a Proposition. However, in many applications of LKIF the attitude of the involved agents towards a proposition will not be relevant at all. For instance, fraud detection applications will only care to distinguish between potentially contradictory observations or expectations relating to the same propositional content. Examples of propositional attitudes are Belief, Intention, and Desire. Each is a component of a mental model, held by an Agent.

Communicated attitudes are held towards expressions: propositions which are externalised through some medium. Statement, Declaration, and Assertion are expressions communicated by one agent to one or more other agents. This classification is loosely based on Searle (1995). A prototypical example of a medium in a legal setting is e.g. the Document as a bearer of legally binding (normative) statements.

When propositions are used in reasoning they have an epistemic role, e.g. as Assumption, Cause, Expectation, Observation, Reason, Fact etc. The role a proposition plays within reasoning is dependent not only on the kind of reasoning, but also the level of trust as to the validity of the proposition, and the position in which it occurs (e.g. hypothesis vs. conclusion). In this respect, the expression module is intentionally left under-defined as a rigourous definition of epistemic roles would include their use in epistemological frameworks used in reasoning (see Section 5.5.2). As a consequence, the ontology allows one to ascribe an epistemic role to some entity, but cannot be used to infer it.

[24] Regardless of whether it is a psychologically plausible account of decision making. Dennett's notion of the *Intentional Stance* is interesting in this context (Dennett, 1987). Agents may do no more than occasionally apply the stance they adopt in assessing the actions of others to themselves.

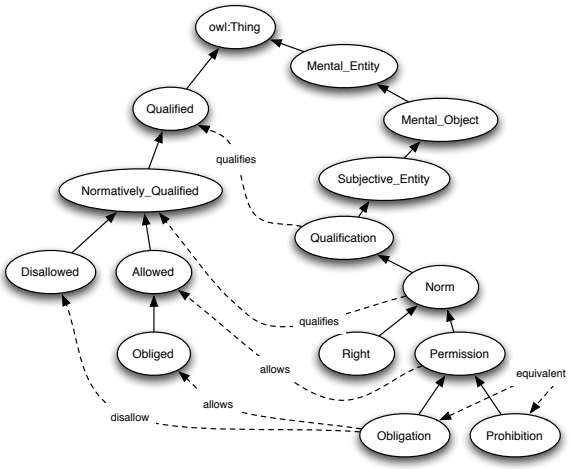


Figure 6.8: Qualifications and Norms

Evaluative_Attitudes express an evaluation of a proposition with respect to one or more other propositions, they express e.g. an evaluation, a value statement, value judgement, evaluative concept, etc. These attitudes express only the type of qualification which is an attitude towards the thing being evaluated, and not for instance the redness of a rose, as in Gangemi et al. (2002) and others. Of special interest is the Qualification, which is used to define norms as specified by Boer et al. (2005a). Analogous to the evaluative attitude, a qualification expresses a judgement. However, the subject of this judgement need not be a proposition, but can be any complex description (e.g. a situation).

### 6.3.3 The Legal Level

Legally relevant statements are created through public acts by both natural and legal persons. The legal status of the statement is dependent on both the kind of agent creating the statement, i.e. Natural_Person vs. a Legislative_Body, and the rights and powers attributed to the agent through mandates, assignments and delegations. At the legal level, the LKIF ontology introduces a comprehensive set of legal agents and actions, rights and powers (a modified version of Sartor (2006); Rubino et al. (2006)), typical legal roles, and concept definitions which allow us to express normative statements as defined in Boer et al. (2005a); Boer (2006); Boer et al. (2007a); Boer (2009).

The Norm is a *statement* that combines two meanings: it is *deontic*, in the sense that it is a qualification of the moral or legal acceptability of some thing, and it is *directive* in the sense that it commits the speaker to ensure that the addressee of the norm complies with it, e.g. through a sanction (Nuyts et al., 2005). These meanings do not have to occur together as it is perfectly possible to attach a moral qualification to something without directing anyone, and it is equally possible to issue a directive based on another reason than a moral or legal qualification (e.g. a warning).

The normatively qualified situations of LKIF are analogous to the generic

situations, or generic cases of Valente (1995). A norm applies to (or qualifies) a certain situation (the Qualified situation), allows a certain situation – the Obliged or Allowed situation – and disallows a certain situation – the Prohibited or Disallowed situation, see Figure 6.8. Allowed and disallowed situations are subclasses of the situation qualified by the norm. Furthermore, they by definition form a complete partition of this situation: a norm applies only to situations that are either mandated or prohibited.

As a consequence, norms of type Prohibition and Obligation are equivalent because they are simply two alternate ways to put the same thing into words: a prohibition to smoke is an obligation not to smoke. A Permission is different in that it allows a situation without prohibiting anything. In that case the logical complement of the allowed situation is some opposite qualified situation about which we only know that it is not obliged.

Where in other approaches, cf. CLO (Gangemi et al., 2005), a situation is the reification of some state of affairs, the normatively qualified situations in LKIF Core are *instantiated* by states of affairs: they are defined as class descriptions that represent a set of possible states of affairs. This means that a standard reasoner can infer whether some actual situation is subsumed under a generic situation, and thus whether norms exist that allows or disallows that situation. Similarly, a classifier will create an inferred hierarchy of situations, which enables a relatively straightforward resolution of *lex specialis* exceptions between norms (Boer et al., 2005a).

## 6.4 Discussion

This chapter presented a principled approach to ontology development in the legal domain, and described how it was applied in the development of the LKIF Core ontology. The main requirements for this knowledge representation ontology, as part of LKIF, are that it should be a resource for special, legal inference, and that it provides support for knowledge acquisition. The legal domain poses additional requirements on the ontology in that it presupposes a common sense grounding. Section 6.2.1 discusses several of the types of ontologies introduced in Section 5.4.1 and argues where they fall short in providing this grounding. Section 6.2.2 presents an approach based on insights from cognitive science and knowledge representation. This approach was incorporated in the middle-out methodology applied in the development of the ontology. The modules and main concepts of the LKIF Core ontology are discussed in Section 6.3. LKIF Core was represented using the OWL 2 DL language described in Chapter 3, and is available at the `http://purl.org/net/lkif-core` persistent URL.[25]

As LKIF Core was developed by a heterogeneous group of people, we specified a number of conventions to uphold during the representation of terms identified (Breuker et al., 2007). One of these is that classes should be represented using necessary & sufficient conditions as much as possible (i.e. by means of owl:equivalentClass statements). Using such 'complete' class definitions en-

[25]The home page for the ontology is currently hosted at `http://www.estrellaproject.org/lkif-core`. Separate modules of the ontology are reachable by adding the module name at the end of the persistent URL, e.g. `http://purl.org/net/lkif-core/action` for the action module.

sures the ability to infer the type of individuals. In retrospect, the extensive use of a combination of Generic Concept Inclusion axioms (equivalent class statements on existential restrictions) and inverse properties turned out to be quite taxing for (in our case) the Pellet reasoner. As at this time we used the reasoner primarily for debugging purposes, a single inconsistency in the TBox could cause the reasoner to stall, making it hard to debug the ontology. Although this problem was remedied by lifting some of the restrictions on classes in LKIF Core, it indicates that real time performance of reasoners on ontologies that use the full expressiveness of OWL 2 DL can still be improved.[26] The discussion of design patterns in the next chapter proposes a way to improve reusability by creating *summaries* of complex class definition patterns (see Section 7.3).

Using a large ontology such as LKIF Core in practice will inevitably be taxing in other ways as well (see Section 5.4). A reusing ontology will need to ascribe to its entire set of ontological commitments. Critical users may have trouble reconciling their own views with the ontology, where naive use may overlook important consequences of this commitment. Furthermore, the ontology involves a level of granularity and detail that may not directly fit a domain ontology. This can be overcome by importing available, compatible domain ontologies in a representation based on the LKIF ontology, provided that safe reuse is ascertained. A possibility is to create library of guaranteed compatible ontologies as part of the LKIF architecture. Reusing a core ontology requires a significant investment, even though improving reusability and facilitating knowledge acquisition are some of its principal purposes. For a knowledge representation ontology such as LKIF Core, the proof of the pudding is in the eating: the pay-off of adopting the ontology in reasoning should exceed the investment it requires.

The LKIF ontology has been successfully applied in the development of a generic architecture for legal reasoning (HARNESS), as the main knowledge resource for performing normative assessment.[27] Hoekstra et al. (2007) evaluates the LKIF definition of Norm in the representation of part of an EU directive on traffic regulations. In van de Ven et al. (2008b,a) we adopted the definitions of the norms module to drive a Protégé plugin for normative assessment. This plugin has been tested to work with representations of both a toy domain of library regulations, and a part of the Hungarian tax law. It uses the Pellet reasoner to compute the subsumption hierarchy of generic cases qualified by the body of norms, and uses this information to generate a *lex specialis* exception hierarchy of norms.

It turned out that extending the ontology with concepts required for the domain of the regulations was relatively straightforward. However, specifying the contents of norms in terms of OWL constructs remained an arduous task because of the sheer complexity of the situations they govern. This experience suggests that the ontology should be extended with a module for expressing measurements, and led to the specification of design patterns that allow us to adequately capture *transactions* and other complex concepts (Hoekstra and Breuker (2008) and Section 7.2). Klarman et al. (2008) propose a way to combine the LKIF ontology with a versioning scheme that corresponds to a specification

[26] Thanks to Taowei David Wang for pointing this out, see `http://lists.owldl.com/pipermail/pellet-users/2007-February/001263.html`

[27] HARNESS: Hybrid Architecture for Reasoning with Norms by Employing Semantic Web Standards

of validity intervals for legal sources.

The following chapter gives an overview of several design patterns identified in the construction of the LKIF ontology. The purpose of this overview is twofold. First, and foremost, it illustrates the wide applicability of design patterns across domains, and secondly, it provides insight in the definition of some of the central concepts in the LKIF ontology.

# Chapter 7

# Design Patterns

"Art is the imposing of a pattern on experience, and our aesthetic enjoyment is recognition of the pattern."

*Alfred North Whitehead, Dialogues (1954)*

## 7.1 Introduction

This chapter is an investigation in the applicability and reuse of structured ontology design patterns as defined in Section 5.6.4. Experience in the development of the LKIF Core ontology has shown that constructing a large ontology in a middle out fashion facilitates the emergence of such patterns. This is not just due to the fact that practice and the subsequent and separate representation of clusters of basic concepts makes one aware of recurrent structures. But the strong methodological emphasis on *reuse* of existing definitions targets awareness of the *similarities*, rather than *dissimilarities* between clusters. Finally, the commonsense and knowledge representation perspectives of the preceding chapter allow the ontology engineer some distance from the theoretical intricacies of the domain being represented.

The discussion of patterns that emerged in the construction of LKIF Core has several purposes. It provides a description and discussion of common design patterns for the purposes of reuse. At the same time, it is a more in-depth introduction to several important notions in LKIF Core, such as subjective, social concepts on the one hand and physical concepts such as processes on the other. Secondly, the reusability of patterns exemplified by these notions are evaluated by applying them to diverse use cases.

A third consideration is that a thorough description of the application of these design patterns gives insight in the way OWL 2 constructs are applied in the construction of complex class descriptions. As we have seen in Chapter 5

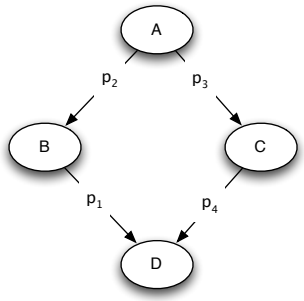


Figure 7.1: Diamond-shaped class description

the methodological and theoretical guidelines that apply to ontology development rarely if ever touch on the *use* of a knowledge representation language. Examples and guidelines for the application of OWL are reserved to either tutorials for novice users (Horridge et al., 2007), or an enumeration of common misunderstandings targeted to improve such educational resources (Rector et al., 2004). For sure, these efforts improve both the understanding of the language, and the ability to assess whether what is represented is an adequate representation of the domain, but they leave open the intermediate step: the representation *process* itself.

There are some examples of more practical 'cookbook' approaches, such as Seidenberg and Rector (2006)'s discussion of transitive properties in OWL, and work on practical structured design patterns such as in Noy and Rector (2006); Bobillo et al. (2007). Similarly, Gangemi (2005); Presutti et al. (2008) focus mainly on *content* patterns discussed in Section 5.6.4. However, these are often problem and domain specific, and present the design pattern as a given, readily reusable ontology modules. Furthermore, considering that design is a task, a *process*, it seems odd to describe design patterns only in terms of declarative structures, rather than by giving the procedure for applying them. Especially since the most valuable contribution of design patterns lies with the possibility to reuse and recombine them to form representations of new, uncharted domains.

### 7.1.1 Dealing with Models

The development of a large ontology such as LKIF Core is not only complex at a conceptual level, but involves numerous encounters with the limitations of the knowledge representation language as well. In some cases the combination of highly expressive language features is such that reasoner performance is severely reduced (cf. the GCI problem in Section 6.4), and sometimes the expressiveness required surpasses that of OWL 2 DL. The latter problem is fundamental, because the language puts a *hard* limit on what you can and can't express. Unfortunately, the limits of OWL 2 DL can suddenly pop up in the most unexpected places, and OWL representations often appear to be in good shape until a seemingly minor change brings the whole thing to shambles.

The semantics of a DL knowledge representation language defines a space

| Class Membership | Property Assertions | Model of **A** |
|---|---|---|
| $a \in \mathsf{A} \sqcap \mathsf{B} \sqcap \mathsf{C} \sqcap \mathsf{D}$ | $a$ related to itself | 7.2a |
| $a \in \mathsf{A} \sqcap \mathsf{D}$ and $b \in \mathsf{B} \sqcap \mathsf{C}$ | $\mathsf{p}_2(a,b), \mathsf{p}_3(a,b),$ $\mathsf{p}_1(b,a), \mathsf{p}_4(b,a)$ | 7.2b |
| $a \in \mathsf{A}$, $b \in \mathsf{B} \sqcap \mathsf{C}$ and $d \in \mathsf{D}$ | $\mathsf{p}_2(a,b), \mathsf{p}_3(a,b),$ $\mathsf{p}_1(b,d), \mathsf{p}_4(b,d)$ | 7.2c |
| $a \in \mathsf{A}$, $b \in \mathsf{B}$ and $d', d'' \in \mathsf{D}$ | $\mathsf{p}_2(a,b), \mathsf{p}_3(a,b),$ $\mathsf{p}_1(b,d'), \mathsf{p}_4(b,d'')$ | 7.2d |
| $a \in \mathsf{A} \sqcap \mathsf{B} \sqcap \mathsf{C}$ and $d \in \mathsf{D}$ | $\mathsf{p}_2(a,a), \mathsf{p}_3(a,a),$ $\mathsf{p}_1(a,d), \mathsf{p}_4(a,d)$ | 7.2e |
| $a \in \mathsf{A} \sqcap \mathsf{B} \sqcap \mathsf{C}$ and $d', d'' \in \mathsf{D}$ | $\mathsf{p}_2(a,a), \mathsf{p}_3(a,a),$ $\mathsf{p}_1(a,d'), \mathsf{p}_4(a,d'')$ | 7.2f |
| $a \in \mathsf{A} \sqcap \mathsf{C}$, $b \in \mathsf{B}$ and $d \in \mathsf{D}$ | $\mathsf{p}_2(a,b), \mathsf{p}_3(a,a),$ $\mathsf{p}_1(b,d), \mathsf{p}_4(a,d)$ | 7.2g |
| $a \in \mathsf{A}, b \in \mathsf{B}, c \in \mathsf{C}$ and $d, d', d'' \in \mathsf{D}$ | $\mathsf{p}_2(a,b), \mathsf{p}_3(a,c),$ $\mathsf{p}_1(b,d'), \mathsf{p}_4(c,d'')$ | 7.2h |
| $a \in \mathsf{A}, b \in \mathsf{B}, c \in \mathsf{C}$ and $d, d', d'' \in \mathsf{D}$ | $\mathsf{p}_2(a,b), \mathsf{p}_3(a,c),$ $\mathsf{p}_1(b,d), \mathsf{p}_4(c,d)$ | 7.2i |

Table 7.1: Valid models of the class A as depicted by figures 7.2a – 7.2i.

of describable valid models (Section 2.5.1). Given the set of language primitives provided by the language, we can go about carving up that space by ruling out certain models. For example, an owl:disjointWith relation between two classes A and B rules out all models where some individual *i* is a member of both. In general terms, the problem is that the surface structure of class level descriptions does not correspond to the structure of allowed models. There are no heuristics in place that assist a knowledge engineer in determining what primitives to use when.

As illustration, consider an ontology $\mathcal{O}$ with primitive classes A, B, C and D. We introduce roles $\mathsf{p}_1, \mathsf{p}_2, \mathsf{p}_3$ and $\mathsf{p}_4$ with no domain or range defined, and give the following description of class A:

$$\begin{aligned} \mathsf{A} \quad \equiv \quad & \mathsf{p}_2 \textbf{ some } (\mathsf{B} \sqcap \mathsf{p}_1 \textbf{ some } \mathsf{D}) \\ & \sqcap \mathsf{p}_3 \textbf{ some } (\mathsf{C} \sqcap \mathsf{p}_4 \textbf{ some } \mathsf{D}) \end{aligned}$$

At the *class* level, this definition of A has a structure that resembles the shape of a diamond (see figure Figure 7.1), but this structure is not reflected at the *instance* level. When we assert the individuals $a, b, c, d, d'$ and $d''$, the definition of A may allow (at least) all patterns of relations between these individuals depicted in Figure 7.2, given the assertions of Table 7.1.

Conversely, we might want to achieve a pattern similar to figure 7.2i at the *instance* level, using a much simpler structure at the *class* level. This is the case when one or more of the classes A, B, C or D are the same, or when some of the roles $\mathsf{p}_1, \mathsf{p}_2, \mathsf{p}_3$ and $\mathsf{p}_4$ are equivalent. Achieving a representation of A that adequately covers the intended structure, comes down to finding ways to either prevent or ensure the possibility of a single individual to play multiple roles. It requires us to prune the set of possible models that satisfy the class description and achieve complete *lockdown* on unintended interpretations. As with a complicated phase in a game of Twister, the slightest slip can have significant

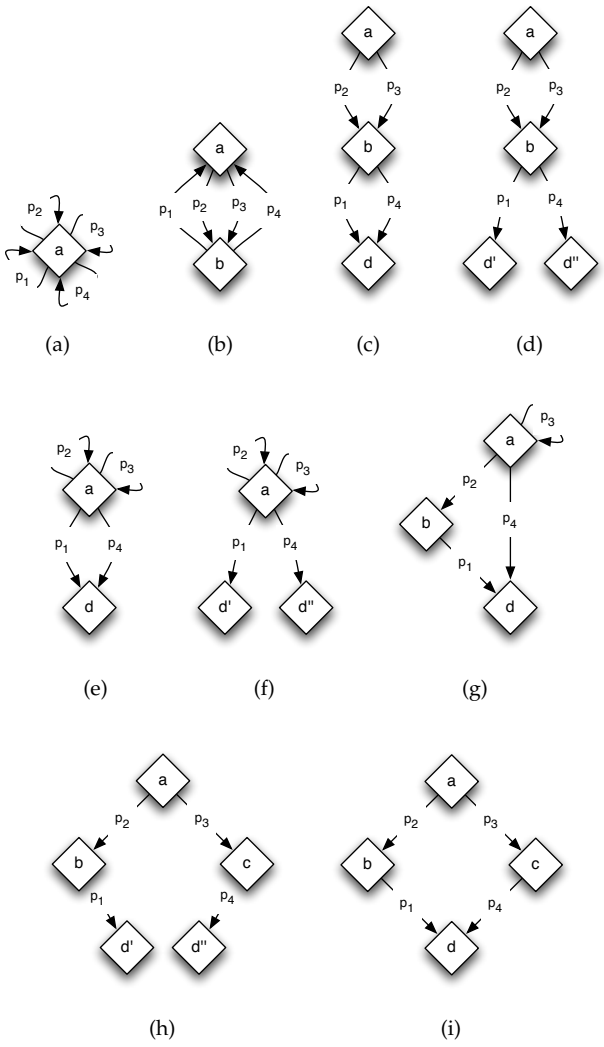


Figure 7.2: Valid models of class A

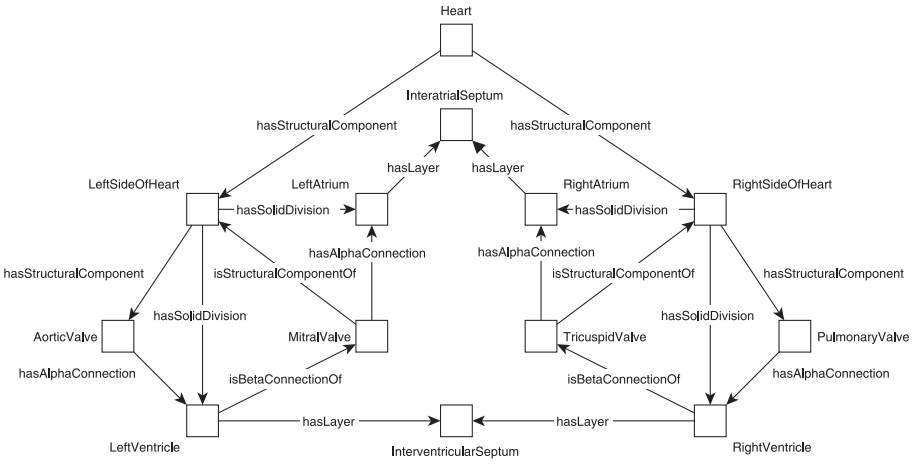


Figure 7.3: Structured object: the heart (Motik et al., 2007a)

consequences.

Fortunately, there is significant progress with respect to automatic debugging and explanation facilities in e.g. Pellet and Protégé 4 based on belief revision (Halaschek-Wiener et al., 2006). The Tweezers extension of Pellet, described in Wang and Parsia (2007), allows an engineer to examine the models generated by its tableaux algorithm.[1]

### 7.1.2 Structured Concepts

Arguably, the limits of expressiveness become increasingly more tangible as the complexity of the domain increases. In fact, many relatively simple domains involve structures that are not necessarily complex by themselves, but *do* push the limits of description logics and consequently result in intricate representation. For instance, the physical structure of organs, molecules, artefacts, devices, but also the interaction patterns between actions, processes and their participants are complex combinations of interdependent entities that are hard, if not impossible to represent in DL.

Motik et al. (2007a) emphasise the need to be able to describe *structured objects*, objects that are defined primarily by the (internal) composition of their parts. A good example, also employed by Motik et al. is the human heart (see Figure 7.3). The heart consists of four 'chambers', the left and right atria, and the left and right ventricles, each of these are separated by a septum, the interatrial and ventricular septa, respectively. However, in OWL DL it is impossible to express that e.g. the atria are separated by the *same* septum.

In SUMO, complex concepts are represented as rules in Common Logic (ISO/IEC, 2007), a highly expressive but undecidable language. This allows one to represent the reciprocity constraints in a direct way, by means of variables. As discussed by Grosof et al. (2003), decidable Horn logic programs

[1] See `http://www.mindswap.org/~tw7/work/profiling/code/`, and its recent incarnation as a Protégé 4 plugin, SuperModel, at `http://www.cs.man.ac.uk/~bauerj/supermodel/`.

allow the definition of non tree-like structures in a straightforward manner. For instance, a definition of the class A, based on the pattern of Figure 7.1 can be captured by the following rule:

$$\begin{aligned} \mathsf{A}(?a) \quad \leftarrow \quad & \mathsf{p}_2(?a, ?c) \wedge \mathsf{p}_3(?a, ?b) \wedge \mathsf{C}(?c) \wedge \mathsf{B}(?b) \\ & \wedge \mathsf{p}_1(?c, ?d) \wedge \mathsf{p}_4(?b, ?d) \wedge \mathsf{D}(?d) \end{aligned}$$

At first sight, it may seem beneficial to adopt an approach that combines a rule-based formalism with OWL 2 DL. However, this combination has several drawbacks. First of all, rules bring back the epistemological promiscuity and order dependence of control knowledge. Furthermore, like the procedural and interpretive attachments of e.g. KRL and KL-ONE (Bobrow and Winograd, 1976; Brachman, 1979, and Section 2.2.5), they can only *assert* new knowledge. Rules cannot aid in posing extra restrictions for classification: they cannot *prevent* classification in the way that integrity constraints in databases do (Hoekstra et al., 2006). Although Motik et al. (2007b) and others have made progress in combining DL with integrity constraints, these developments have not as yet found their way into applications.

Maintaining consistency between inferences of the classifier and the rule interpreter is not trivial. This is largely a consequence of the safeness condition necessary for decidable combinations of rule and DL formalisms (Motik et al., 2005). Because variables in DL-safe rules are only allowed to be bound to known, *named* individuals, a rule interpreter will not take the possibly infinite number of anonymous individuals inferred by an OWL 2 DL reasoner into account: rules are likely to quantify over incomplete models of the TBox.

Hybrid approaches that reconcile DL with rules, such as DLP (Grosof et al., 2003), Horn-$\mathcal{SHIQ}$ (Krötzsch et al., 2006), and the recent ELP (Krötzsch et al., 2008b) and $\mathcal{SROIQ}$-Rules (Krötzsch et al., 2008c) inevitably sacrifice expressive power to constrain reasoning complexity. For instance, DLP is defined as the intersection of DL and logic programming languages,[2] the ELP language requires variables in the body of a rule to be connected either via a tree-shaped graph or a set of such graphs, the $\mathcal{SROIQ}$-Rules language is restricted to those rule-like constructs expressible in the $\mathcal{SROIQ}$ DL. In short, languages either provide expressive rules with limited DL, or expressive DL with limited rules. As a consequence, complex concepts are often defined using a language of the former form, even though this is not always necessary. The question remains: what *can* we still meaningfully express in OWL 2 DL?

### 7.1.3 Three Patterns

This chapter discusses OWL 2 DL representations of three frequently occurring patterns. Rather than full fledged class definitions, these patterns are presented by the way they are assembled using the OWL 2 primitives discussed in Chapter 3. In this description, the various design decisions and limitations of the representation are discussed in detail, and for each pattern it is shown how it can be applied to various domains.

The first pattern concerns the *diamond* structure briefly discussed in the preceding section. Section 7.2 is an extended version of Hoekstra and Breuker

[2] This approach lies at the heart of the OWL 2 RL profile of OWL 2. See Section 3.5.2.

(2008), and investigates ways to approximate this structure using some of the more expressive language features of OWL 2 DL. Central to this pattern is the ability to infer identity relations between different participants in the pattern.

The second pattern is concerned with the representation of relations and their reifications. Section 7.3 discusses how several use cases, such as n-ary relations and subjective entities, can be reduced to the problem of inferring a relation between two individuals that participate in the reification of that relation. The pattern describes *triangular* structures, which can be combined to define complex (social) relations.

The third pattern allows the representation of *sequences* of multiple entities. Section 7.4 presents the pattern and shows how it can be applied in the definition of causal relations and complex protein structures. The pattern relies heavily on the role inclusion axioms of OWL 2 and is compared to an alternative pattern described by Drummond et al. (2006).

Finally, in Section 7.5 the three patterns are combined in the representation of the Action class of LKIF Core. The wide applicability and reusability of this pattern-based representation shows significant benefits compared to the classical middle-out approach of Section 5.2. Nonetheless, it is quite possible that the patterns discussed here gather their applicability from the basic nature of the concepts that triggered their original representation in the first place.

The patterns discussed in this chapter are representations of notions that stand in the spotlight of theoretical discussion in both philosophy and AI. In line with the knowledge representation perspective of the preceding chapters, these representations are illustrations of the capabilities and usability of OWL 2 DL in a variety of domains and make no claim with respect to ontological, nor theoretical correctness. Knowledge representation is a means to an end, and we can only aim to *approximate* reality.

## 7.2 Grasping the Diamond: *The Reciprocity of Exchange*

Concepts in general ontologies are basic building blocks for the definition of more specific and compound notions. They are not only domain independent and generally applicable, they are necessarily also abstract and simple. In various papers we have advocated a crisp distinction between basic ontological categories, and more complex notions, captured in *frameworks* (Breuker and Hoekstra, 2004c; Hoekstra et al., 2007, and Section 5.5.2). Although this is the general rule, some compound concepts are of a clear ontological nature. Construction of the LKIF Core ontology prompted us to consider a term frequently occurring in law – e.g. in contracts – that is both very abstract, and composite: the *transaction*. Transactions bind two actions by *reciprocity*, which is particularly reflected by identity and other constraints on roles in the two constituting actions. It forms a graph pattern that is diamond-shaped (see Figure 7.4).

A generic use case (cf. Gangemi (2005)) that covers these reciprocal combinations is the notion of *exchange*. Exchange not only pervades the full range of physical, mental and abstract changes, but also takes a very abstract position in ontologies such as LKIF-Core and SUMO: it is an unavoidable concept and appears in various forms, e.g. in the definition of financial transaction of

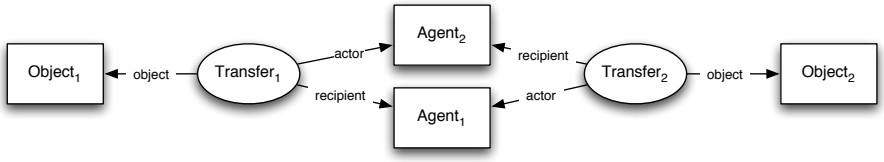


Figure 7.4: Structure of a transaction

SUMO (Niles and Pease, 2001). Entities such as objects and processes can undergo any of three kinds of change. In a *change of value*, the value of a property of an entity changes: e.g. a traffic light changes to green; a car accelerates. *Transfer* is a change of a value of a (anti-rigid) property that is not inherent to the entity: e.g. an object is moved to another position, a person graduates. Usually the property that changes is a position. A *transformation* is the change of a rigid property: e.g. a solid physical object melts to form a liquid. All three types of change can occur in the context of an exchange. A trivial example of ex-'change-of-value' is changing currencies. Metabolism is an example of an exchange consisting of transformations where material and energy are exchanged between an organism and its environment.

However, exchange involves additional constraints with regard to reciprocity. For instance, a sales transaction consists of two transfer actions: a buyer receives some goods for which he pays money, and the seller receives that *same* money and provides the *same* goods (Figure 7.4). For any exchange, the reciprocity constraints are:[3]

**Identity**
the active participants in both transfers are the same. For instance, the agent that is the actor of the money transfer (paying) is also the receiver of the goods. The agent that is the actor that transfers the goods, i.e. the seller, is the receiver of the money.

**Balance**
the objects in both transfers, i.e. the things being exchanged, should be comparable in some way. For instance, the value of the money should be balanced by the value of the goods.

A position metaphor can be applied to the participants of an exchange, e.g. goods and money are said to *trade places*. The same metaphor holds in a legal context for the exchange of ownership: ownership is a *legal* position, a right Hohfeld (1919). Again, legal positions themselves involve reciprocity as well: the right to own is balanced by a duty to pay.

Although indeed this section focuses on *exchange* as primary use case, similar structures can be found in more static notions such as physical (situation) descriptions, biological structures (Motik et al., 2009), dependency diagrams, and more theoretical notions such as Hohfeldian squares (Hohfeld, 1919).

In the next section, the representation of transactions is described in a step-by-step fashion, showing at the same time the practical knowledge pattern,

[3]Observe that the words 'buying' and 'selling' refer to the same conceptual entity. They are not synonymous; the choice of word is determined by the linguistic-pragmatic context.

and a hands-on methodology for approximating diamond-like structures using OWL 2 DL primitives.

### 7.2.1 Representing Transaction

A knowledgeable user of OWL will most likely only encounter real problems when dealing with elaborate, highly structured concepts. Often these problems are so big, or just insurmountable, that we resort to a rule-based representation. For the reasons discussed in Section 7.1.1, this is undesirable or at least should be avoided when possible. Only when the expressive power of either one of the formalisms is pushed to its limits, problematic interactions between the two formalisms can be avoided.

Just like the human heart, used as example by Motik et al. (2007a), the notion of transaction is a structured concept. The following sections take this use case to elaborate on the tools we can use to eliminate some of the undesirable models allowed by an initial, naive representation. The purpose of the final representation is twofold:

**Identification**
to be able to correctly *recognise* an individual transaction as a member of a transaction class, but also

**Enforcement**
to generate a violation when an individual is incorrectly asserted to be a member of that class.

The methodology consists of a succession of five steps:

#### Step 1: Initial Class Definition

The straightforward definition of a transaction is a Transaction class that has transfer actions Transfer as parts. In turn, each Transfer has an actor and recipient property with some agent Agent as value, and an object relation with an object Object:

$$\begin{aligned} \mathsf{Transaction} &\equiv \mathsf{part}\ \textbf{some}\ \mathsf{Transfer} \\ \mathsf{Transfer} &\equiv \mathsf{actor}\ \textbf{some}\ \mathsf{Agent} \sqcap \mathsf{recipient}\ \textbf{some}\ \mathsf{Agent} \\ &\quad \sqcap \mathsf{object}\ \textbf{some}\ \mathsf{Object} \end{aligned}$$

#### Step 2: Constrain the Number of Role Fillers

Evidently, the above definition only prescribes that there must be at least one part relation with a Transfer. We can fine tune this definition by specifying that a Transaction has only instances of Transfer as its parts and provide a cardinality restriction on the number of part relations allowed. However, this is too strict as it no longer allows for any other part relations either. In OWL DL 1.0 the only way out of this predicament was to introduce a specific has_transfer sub property of part for the Transaction class. For an individual to be inferred to be a Transaction, it should be expressed using the (class specific) has_transfer relation. However, using the qualified cardinality restrictions of OWL 2 DL

we can specify that a Transaction should have exactly two part relations to a Transfer individual:

$$\text{Transaction} \sqsubseteq \text{part } \textbf{exactly } 2 \text{ Transfer}$$

Identification of individuals can only be achieved by means of equivalence axioms in combination with the domain and range of a property. Some types of restrictions cannot be used for identification. Because of the open world assumption, an upper bound on the cardinality of a property at some class can not be used to *identify* an individual to be a member of that class, as there is no way to know whether the individual has additional properties of that type. For the same reason, universal restrictions cannot be used for identification using equivalence axioms.

Once the type of an individual is established, additional subclass axioms can be used to infer new properties of the individual: it is shaped to conform with the pattern expressed in the restriction. Of course, this approach can be problematic when the equivalence axiom is under specified, as too many individuals may be inferred to be a member of that class. Using this methodology, we can refine the Transaction class as follows:

$$\begin{aligned} \text{Transaction} &\equiv \text{part } \textbf{some } \text{Transfer} \sqcap \text{part } \textbf{min } 2 \text{ Transfer} \\ &\sqsubseteq \text{part } \textbf{only } \text{Transfer} \sqcap \text{part } \textbf{exactly } 2 \text{ Transfer} \end{aligned}$$

Admittedly, this definition of Transaction is quite strict, as it does not allow for anything having more than two Transfers as part. For any individual that does have more than two, the reasoner will infer some of the Transfer individuals to be the same. Asserting that these individuals are mutually different, will make the ABox incoherent. Of course, we can overcome this limitation by refining the definition of Transfer to include only those transfer actions taking part in a proper transaction. However, simply stating that every Transfer should be part of a transaction does not add any new information. Nonetheless, requiring that any Transaction should have (at least) two individual Transfers as part ensures that the definition is not 'triggered' for individuals that are not asserted to be mutually distinct.[4]

#### Step 3: Disambiguate Role Fillers

Let's turn our attention to Transfer; we restrict it in a similar fashion:[5]

$$\begin{aligned} \text{Transfer} &\equiv \text{actor } \textbf{some } \text{Agent} \sqcap \text{recipient } \textbf{some } \text{Agent} \\ &\quad \sqcap \text{object } \textbf{some } \text{Object} \\ &\sqsubseteq \text{actor } \textbf{exactly } 1 \text{ Agent} \sqcap \text{recipient } \textbf{exactly } 1 \text{ Agent} \\ &\quad \sqcap \text{object } \textbf{exactly } 1 \text{ Object} \end{aligned}$$

In fact, the definition of Transfer adds complexity because it involves *distinct properties* and different *ranges*. However, this heterogeneity allows us to drop

[4]Note that the distinctness of two individuals cannot be enforced by means of a class description, as the open world assumption allows for the existence of some other (unspecified) distinct individual.

[5]For the current purposes, the participant properties could be made *functional* thereby waiving the necessity for qualified cardinality conditions. However, functionality is a rather strong claim as it is a global restriction whereas cardinality restrictions operate locally on a class and its subclasses.

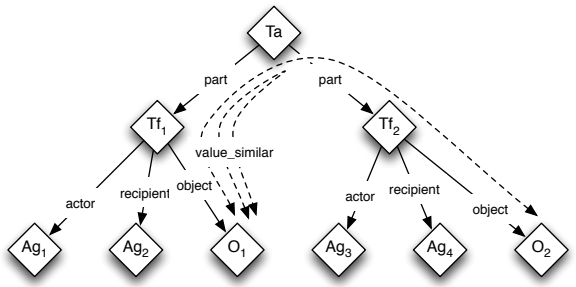


Figure 7.5: Tree-model as described by the definitions of Transaction and Transfer. The dotted arrows depict the path along the tree by the property 'value_similar'.

the owl:allValuesFrom axiom as the combination of exact cardinality and existential restriction has the same effect. The complexity resides in that we need a way to distinguish the actor from the recipient. The above definition does indeed ensure that we have at least an actor and a recipient for any Transfer, but these individuals may still be the same. Asserting disjointness between Agents in different roles is not desirable, as this eliminates all models where each Agent fulfils both roles at the same time. Fortunately, OWL 2 introduces the ability to make the extensions of the properties disjoint.[6]

#### Step 4: Traverse the Tree

Figure 7.5 shows the completion graph for the definitions of Transaction and Transfer we have so far. As one can see it is tree-shaped and e.g. does not require that the actor of transfer $Tf_1$ and the recipient of $Tf_2$, $Ag_1$ and $Ag_4$ respectively, are the same. Because of the tree-model property of $\mathcal{SROIQ}(D)$, we cannot further extend these definitions to enforce such a restrictions. However, we can still *infer* a lot of information that we can use to further restrict the various participants in the transaction.

We first shift our attention to the two Object individuals $O_1$ and $O_2$ appearing in the two transfers. As we discuss in Section 7.2, these two objects need to have a balance in 'value': good vs. money in transactions, kinetic vs. potential energy in collisions, etc. OWL 2 DL property chains specify the propagation of a property along some path of interconnected properties. We introduce the property value_similar that connects the two objects of the transaction as follows:

$$\mathsf{object}^- \circ \mathsf{part}^- \circ \mathsf{part} \circ \mathsf{object} \sqsubseteq \mathsf{value\_similar}$$

Because of the left-right symmetry of the tree-model of Transaction the value_similar property is symmetric and reflexive; because both $O_1$ value_similar $O_2$ and $O_2$ value_similar $O_1$ hold, and the two part relations connecting $Ta$ to $Tf_1$ and $Tf_2$ cannot be distinguished.

[6] i.e. for any two individuals $x$, $y$ and actor$(x, y)$ there exists no recipient$(x, y)$.

We could similarly specify a symmetric and transitive same_id_as property that specifies (in a custom manner) that two individuals share identity:

$$\begin{aligned} \mathsf{actor}^- \circ \mathsf{part}^- \circ \mathsf{part} \circ \mathsf{recipient} &\sqsubseteq \mathsf{same\_id\_as} \\ \mathsf{recipient}^- \circ \mathsf{part}^- \circ \mathsf{part} \circ \mathsf{actor} &\sqsubseteq \mathsf{same\_id\_as} \end{aligned}$$

#### Step 5: Introduce Asymmetry

Unfortunately, this chain is overly general as it infers same_id_as relations between the actor and recipient participating in a single action. Again, this is caused by the symmetry of the two part branches of the completion graph of Transaction. As in OWL 2, complex properties such as property chains cannot be disjoint, the only way to rule out models in which the two participants are inferred to be the same, is to make the tree itself asymmetric. We can do this in two ways.

Firstly, we could make the generic definition of Transaction intrinsically asymmetric by disambiguating the left-hand side and right-hand side of the tree using two distinct part properties. This has the drawback we had before, i.e. it is rather ad hoc and makes identification tautological. Secondly, rather than enforcing asymmetry, we can exploit available asymmetry by disambiguating the two sides of the tree via the nature of the participants in the two transfers. This solution has the drawback that the participants on both sides need to be suitably distinct, and that sameness can only be expressed in terms of the characteristics of those participants. In other words, it increases domain dependence of the representation: the same_id_as relation can not be inferred for the generic Transaction class.

Granted that the reader agrees that the first disadvantage outweighs the second, we continue to disambiguate between different participants. Suppose our transaction is a sales transaction where some Good is exchanged for a quantity of Money. We can use this information to discriminate between the two Transfers that make up our Transaction:

$$\mathsf{Goods\_Transfer} \equiv \mathsf{Transfer} \sqcap \mathsf{object}\ \mathbf{some}\ \mathsf{Good}$$

Again, the equivalence axiom is used to grab the Transfer individuals that involve goods. We can then use our strict prior definition of Transfer to enforce object specific relations between the Transfer and the actor and recipient:

$$\begin{aligned} \mathsf{recipient}_g &\sqsubseteq \mathsf{recipient} \\ \mathsf{actor}_g &\sqsubseteq \mathsf{actor} \\ \mathsf{Goods\_Transfer} &\sqsubseteq \mathsf{recipient}_g\ \mathbf{some}\ \mathsf{Agent} \sqcap \mathsf{actor}_g\ \mathbf{some}\ \mathsf{Agent} \end{aligned}$$

The money transfer Money_Transfer and properties can be defined accordingly. Because of the exact qualified cardinality constraints on Transfer for the participants object, actor and recipient, a DL reasoner will infer that for any Goods_Transfer $g$ and Agent $a$, if $g$ actor $a$ then $g$ $\mathsf{actor}_g$ $a$. Based on this inference, we

can rephrase the same_id_as relation in terms of context specific relations:

$$\begin{aligned}
\mathsf{actor}_g^- \circ \mathsf{part}^- \circ \mathsf{part} \circ \mathsf{recipient}_m &\sqsubseteq \mathsf{same_id_as} \\
\mathsf{actor}_m^- \circ \mathsf{part}^- \circ \mathsf{part} \circ \mathsf{recipient}_g &\sqsubseteq \mathsf{same_id_as} \\
\mathsf{recipient}_g^- \circ \mathsf{part}^- \circ \mathsf{part} \circ \mathsf{actor}_m &\sqsubseteq \mathsf{same_id_as} \\
\mathsf{recipient}_m^- \circ \mathsf{part}^- \circ \mathsf{part} \circ \mathsf{actor}_g &\sqsubseteq \mathsf{same_id_as}
\end{aligned}$$

Given the transaction $Ta$ depicted in Figure 7.5, and $O_1$ : Money and $O_2$ : Good we can now infer the relations $Ag_1$ same_id_as $Ag_4$ and $Ag_2$ same_id_as $Ag_3$.[7]

### 7.2.2 Discussion

The notion of exchange, and more specifically transaction, represents a commonly recurring pattern in many domains. Although it is a concept primarily characterised by its structure, rather than inherent properties, it has a strong ontological flavour. The reciprocity of the various participants in an exchange is similar to constraints in representations of physical artefacts.

The structure of this complex concept can be approximated by exploiting a combination of the newly introduced language constructs in OWL 2 DL. The previous section describes a general methodology that can be applied to create approximate representations of *any* diamond shaped structure. It is composed of five straightforward steps. Create *initial class definitions* that provide a basic means for identifying individuals. Constrain the *number of role fillers* allowed for members of that class, by combining upper and lower bound cardinality restrictions using both subclass and equivalence restrictions, respectively. *Disambiguate role fillers* by making property extensions disjoint. Define property chains that *traverse the tree* to create (identity) relations between different participants in a transaction (or similar pattern). And finally, introduce *asymmetry* by incorporating contextual information, e.g. by using less generic concepts to define the more general relation.

These steps ensure both that candidate individuals in the ABox are appropriately *identified*, and that inferred property values for those individuals are *enforced*. Explicit violation of such inferences, e.g. by means of a negative object property assertion in OWL 2 DL for the same_id_as property, results in an incoherent ABox. However, the same_id_as and other relations introduced in this pattern are *inferred* on the basis of property chains, but they do not enforce the existence of a chain when they are *asserted*. A chain of properties can only be defined as a sub property of another property, but it cannot be stated to be equivalent: the super property does not *define* the chain. This is because if this were possible, we could specify a recursive property, e.g. $\mathsf{p} \circ \mathsf{r} \equiv \mathsf{r}$, and thereby force the existence of an infinite property chain using an existential or cardinality restriction.[8] Therefore, for the pattern to 'work' – and this holds

[7] The good and money in the example can be enforced to be different individuals by making the Money and Good classes disjoint. If this is undesirable (i.e. because money can be considered a good as well), this does raise the need for a specific Sales_Transaction class that consists only of a Money_Transfer of Money that is no Good. Alternatively, one can introduce a custom different_id_from property along the lines of same_id_as.

[8] Note that this is different from an existential restriction on a transitive property, as transitive properties *may* have an infinite length, but do not *have* to.

for all patterns in this chapter – care has to be taken to distinguish between inferable and 'assertable' properties.

Admittedly, because we can only approximate the diamond shape, the most conspicuous drawback is the need for an alternative to owl:sameAs to express that two individuals are 'the same'. On the other hand, in some cases a custom relation is even to be preferred over the owl:sameAs relation such as when it is used to relate two different states of the same object. For instance, transactions can be conceived of as involving *change*, where the state of affairs before the transaction should be distinguished from the result of the transaction (see Section 7.4). In this case, a same_id_as relation can express a shared identity between logically different individuals, where asserting the owl:sameAs relation would result in an incoherent ABox (Hoekstra et al., 2006).

An added advantage is that the pattern steers clear of problems arising from the overlapping expressiveness of DL and logic programs. To close the diamond, we need only define a single, relatively harmless DL-safe rule that maps the same_id_as property to owl:sameAs. However, such a rule cannot infer that some individual is a participant in a complementary action when the participants of that action are not explicitly asserted. An alternative is to use nominal classes as 'placeholders' on the different individuals in the pattern. For instance, the definition of Transaction could be amended in a way that it enforces all actor role fillers of its Transfer parts have the value of a single individual *transaction_actor*. A match of some individual with the Transaction class description will then infer owl:sameAs relations between the (possibly different) individuals that fill the actor roles in that particular case, and the *transaction_actor*. This is only useful in a controlled environment where the structure of individuals that matches a class only occurs once. When multiple transaction individuals match the class description, their actors will be inferred to be the same as well.

## 7.3 Reification and Summarisation: *Relations and Social Reality*

This section describes a pattern that allows for the *summarisation* of reified relations. It is argued that although a reified relation often conveys a stronger ontological commitment than the original relation, while at the same time the original relation is often more convenient for practical use. The two sides of the coin are discussed in the context of two use cases: the representation of *n-ary relations* (Noy and Rector, 2006) and a description of *social reality* (Searle, 1995).

### 7.3.1 N-Ary Relations

One of the design patterns discussed by the Semantic Web Best Practices and Deployment (SWBP) working group of the W3C concerns the representation of so-called n-ary relations: relations between more than two entities (Noy and Rector, 2006).[9] The need for a 'best practices document' on the representation of n-ary relations in OWL DL originates from two phenomena. Because OWL

[9] See `http://www.w3.org/2001/sw/BestPractices`.

does not provide means to express n-ary relations directly, many other representation languages do have built-in support for these relations: the document addresses a *translation problem* for users coming from a different background. For instance, formal ontologists tend to represent ontologies using expressive languages such as traditional first order logic or other languages that allow the definition of such predicates, cf. CommonLogic (ISO/IEC, 2007). At the other end of the spectrum, the TopicMaps[10] language allows for associations with multiple subjects.

Noy and Rector (2006) discuss three use cases where n-ary relations would be in order:[11]

> **Case 1**
> "Christine has breast tumour with high probability."
>
> This phrase hints at a binary relation between the person *Christine* and diagnosis *Breast_Tumor_Christine* with a qualitative probability value describing this relation as *high*: the binary relation really needs a further argument.
>
> **Case 2**
> "Steve has temperature, which is high, but falling."
>
> In this case, *Steve* has two values for two different aspects of a has_temperature relation: its magnitude is *high* and its trend is *falling*. The two binary properties always go together, and should be represented as one n-ary relation.
>
> **Case 3**
> "John buys a 'Lenny the Lion' book from `http://books.example.com` for $15 as a birthday gift."
>
> Here, there is a single relation in which individual *John*, entity `http://books.example.com` and the book *Lenny_the_Lion* participate. This relation has other components as well such as the purpose (*birthday_gift*) and the amount (*$15*).

The solution proposed by Noy and Rector is to represent such n-ary relations as a class with $n$ properties that *reifies* the relation.[12] For instance, the has_diagnosis relation between *Christine* and *Breast_Tumor_Christine* is represented as an individual of type Diagnosis_Relation with properties value *Breast_Tumor_Christine* and probability *high*. Similarly, *Steve*'s has_temperature is represented as an individual Temperature_Observation, and *John*'s book purchase is represented as a Purchase individual (see Figure 7.21).

Regarding the first case, this solution is similar to the *association class* in class diagrams of UML that directly represents some association (relation) between two classes. The association class itself can then have additional properties, such as in Figure 7.6. Association classes can only be used to reify a *binary* relation where the relation itself has additional properties.

[10] See `http://www.topicmaps.org`

[11] Examples and text taken from `http://www.w3.org/TR/swbp-n-aryRelations/`. Noy and Rector describe a fourth use case where a single object is related to an ordered list of other objects, see Section 7.4.

[12] Noy and Rector (2006) avoid the use of this term because of differing interpretation of the term in TopicMaps and RDF.

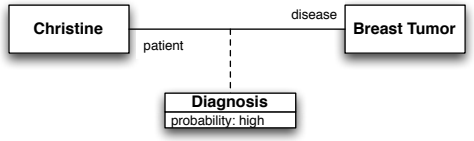


Figure 7.6: The association class 'Diagnosis' in UML, from `http://www.ibm.com/developerworks`

Noy and Rector do not address the fact that these cases are based on the a priori ontological commitment that the individuals involved are part of a *single relation*. How does such a commitment come about? In knowledge representation, and ontology construction in particular, there exists a rather prominent tradition that advocates isomorphism to natural language. This tradition is in part upheld by those having a background in natural language understanding and semantics, whose primary interest is to represent exactly what is expressed in a sentence. A stronger influence is the philosophical stance in ontology that the categories of existence lie at the basis of human cognition. It is furthermore argued in a semiotic tradition, that human cognition, or rather, the categories of human thought, are best conveyed through studying natural language (cf. the discussion of the DOLCE ontology in Section 6.2.1).

The decision to represent some description as a relation is often guided by the common rule of thumb that nouns (or noun phrases) represent concepts where verbs represent relations. Additionally, nouns preceded by 'has', as e.g. 'has temperature' are usually represented as properties as they indicate descriptions that cannot stand on their own: a temperature is always the temperature *of something* (Brachman et al., 1991; Sowa, 2000). A description preceded by a proposition such as 'with' is a modifier of some relation. Adjectives such as 'high' indicate values of relations.

It is these kinds of simple rules that are exploited by controlled natural languages for OWL such as Attempto Controlled English (Kaljurand and Fuchs, 2007, ACE), the Sidney OWL Syntax (Cregan et al., 2007) and Rabbit (Schwitter et al., 2008). There is a reason that these languages are *controlled*: natural language is itself highly mouldable. Consider for instance the case of Christine in Table 7.2. These examples all describe approximately the same thing; they are paraphrases, and casual reading of these sentences seems to convey the same information. However, when we adopt the rules of Brachman et al. (1991), the possible resulting models may become strikingly different for each of the sentences.

The gist of this exercise is to show that the need for a language construct, such as n-ary relations should be based on a conscious decision to interpret a use case in a particular way: it is an ontological commitment. A too direct connection to the way information is conveyed in linguistic expressions may introduce a commitment for cases where the conceptualisation underlying the expression makes no such commitment. Conversely, the proposed alternative representations of cases 1-3 make explicit two additional ontological commitments:

- a commitment to the existence of some intermediate entity through which

1. *"Christine has breast tumour with high probability"*
   Some unspecified n-ary relation between Christine and breast tumour, where the probability of this relation is high;
2. *"Christine has a high probability of breast tumour"*
   A binary relation 'has high probability' between Christine and breast tumour;
3. *"Christine has a diagnosis of breast tumour with high probability"*
   Some binary relation between Christine and diagnosis, and two relations between diagnosis and breast tumour and high probability, respectively;
4. *"Christine is diagnosed as having breast tumour with high probability"*
   An binary relation 'is diagnosed' between Christine and breast tumour, and a probability of this relation with value high;
5. *"Christine's diagnosis is breast tumour with high probability"*
   Some binary relation between Christine and diagnosis of type breast tumour where the probability of this diagnosis is high.

Table 7.2: Different wording, different representation?

the various relata of the n-ary relation are connected.

- a commitment to the non-existence of the n-ary relation itself, for the intermediate entity and the n-ary relation cannot exist at the same time. This does not exclude the possible existence of additional relations between the relata of the intermediate entity.

These commitments are not arbitrary: the reification makes explicit some knowledge about the domain which would be left implicit in the n-ary relation. For each of the cases, the reified class cannot be anything other than respectively a diagnosis, observation or purchase. The n-ary relation is a simplification, or summary, of the underlying conceptual structure. The restriction to binary relations in OWL thus enforces a more ontologically concise representation. The drawback, on the other hand, is that exactly because of this commitment, the n-ary relation itself can no longer be represented. Nonetheless, the design pattern described in Section 7.3.3 can reclaim some of the relational character of the reification.

### 7.3.2 Social Reality

A similar case for reification can be made on the basis of the considerations underlying the intentional and legal levels of LKIF Core (discussed in sections 6.3, 6.3.2 and 6.3.3). Remember that in line with the world knowledge in Valente's functional ontology, the legal level is an abstraction of commonsense knowledge. This abstraction is similar to the way in which intentional and functional notions, are generalised over physical phenomena by the design and intentional stance of Dennett (1987). These notions are *social constructs* that can be attributed to, or imposed on brute facts, phenomena the existence of which does not depend on human agreement (Searle, 1995).

According to Searle, institutional facts are constructed by means of *constitutive* and *regulative* rules, rules of the form:

$$X \text{ counts as } Y \text{ in context } C$$

Examples of constitutive and regulative rules are, respectively:

- Bills issued by the Bureau of Engraving and Printing (X) count as money (Y) in the United States (C).
- For the purposes of this law (C), a house boat (X) is deemed to be a house (Y).

Where a regulative rule imposes additional restrictions on something (i.e. a house boat) to create an institutional fact (the house-boat-as-house) which can exist independently from that rule, the constitutive rule determines (in part) the possibility of existence of the institutional fact. Clearly, where regulative rules have a normative character, constitutive rules are definitional. It is therefore not surprising that the *counts-as* relation has received a lot of attention in AI and Law, most recently in Grossi et al. (2005); Grossi (2007).

Although the counts-as relation in constitutive rules attributes the properties of $Y$ to $X$ in a similar fashion as ordinary subsumption, it is very distinct in three ways:

- The counts-as relation is ontologically *subjective* and observer relative, and therefore *contextual*. It only holds in relation to a certain context, within a system of constitutive rules created by (collective) intentionality.
- The counts-as relation limits inheritance, and does not permit *substitutability* of the syntactic elements of rules. For instance, the statement "Money is the root of all evil" in conjunction with the constitutive rule introduced before, does not make that "Bills issued by the Bureau of Engraving and Printing count as the root of all evil in the United States.". This is because the context of the counts-as relation is intensional; the institutional fact cannot be defined by reference to statements outside the context.
- The counts-as relation can be used to connect *anti-rigid* with *rigid* classes (Guarino and Welty, 2004). For example, where "bills issued by the Bureau of Engraving and Printing" is rigid, the "money" class is anti-rigid.

These characteristics of the counts-as relation make that it cannot be represented using the subsumption relation, which is contextless, extensional and subject to ontological restrictions concerning its relata (Section 5.5.1). Social, institutional facts can accumulate in layers, but eventually need to bottom-out in brute facts. This logical priority of brute facts coincides with the layered perspective of Dennett's stances and corresponds with primacy of physical reality in the LKIF Core ontology.

A fourth characteristic of social constructs is that, because of their context dependency, they are often conceived of as a *relation* between the brute fact and its context. A well known example is the duality of roles as relations and classes (Steimann, 2000; Loebe, 2003). Steimann identifies three ways to represent roles, as *named places* in relations, as *specialisation* or *generalisation*, or as *adjunct instances*. Clearly, of these only the first and last conform to the intensional character of the counts-as relation, and only the latter is directly compatible with OWL.

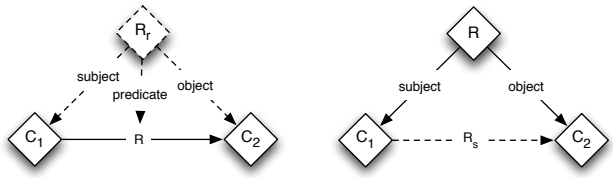


Figure 7.7: Reification versus summarisation of relation R.

Representing roles as adjunct instances means that the role itself and its player are defined as two distinct classes, with a played_by relation. For each occurrence where some entity plays a role, two individuals need to be present in the ABox of a description logic. However, for some purposes this representation may be non intuitive or overly verbose, and the relation between role player, role and context is 'flattened' to a relation between role player and context. This is essentially the named places approach, and although OWL does not support these, a similar effect can be reached by defining the named places as mutually inverse properties.

The role-as-relation approach is enticing in cases where the role-as-class approach involves an apparent circularity. To give an example, it is hard to consider the role Student independently from a student_of relation with a University class. Defining Student in terms of that relation is rather tautological. A similar example is the interplay between a propositional attitude Belief, and the proposition that is believed_by an agent. In the development of LKIF Core, we encountered various other occasions where this pattern emerged (see Section 7.3.4).

### 7.3.3 Representing Summarisation

Both the representation of social reality and n-ary relations involve an important ontological commitment to either a relation oriented approach (impossible in OWL for the n-ary case) or a class centred representation. As we have seen, in many cases the class centred approach is ontologically more concise, but also more verbose. For many applications it is useful, or simply convenient, to abstract away from this ontological commitment. This process of *summarisation* is in fact the reverse of reification. While reification turns a secondary entity into a primary entity – i.e. a relation is reified as a class – summarisation 'hides' a primary entity as a secondary entity. The use cases for n-ary relations of Section 7.3.1 show that a need for representing n-ary relations is usually motivated by relations that already hide an ontological commitment. Arguably, an explicit methodology for creating (or inferring) the summarisation of ontologically concise, but verbose representation can be very useful for e.g. ontology reuse. A reusing ontology may reuse just the abridged version of 'properties' without compromising its ontological commitment.

Figure 7.7 shows the reification $R_r$ of relation R as an individual with explicit subject, predicate and object relations, and the converse summarisation $R_s$ of individual R as a relation between classes $C_1$ and $C_2$. The use of the term 'reification' was discredited by Noy and Rector (2006) because of its different

meaning in different languages (Section 7.3.1). Even when we commit to the single interpretation of reification in RDF, it is clear that the RDF-solution does not meet our needs:

- In RDF, for any property an rdf:Statement can exist that reifies it (Section 3.3.1). However, the converse does not hold: given some rdf:Statement one cannot automatically infer the existence of a property assertion between rdf:subject and rdf:object of the type specified by rdf:predicate. This means that RDF is not expressive enough to support both reification and summarisation using a single mechanism.
- RDF reification is indiscriminate with respect to the type of the relation being reified, whereas the current use cases are restricted to n-ary relations and constitutive rules.
- Though OWL has an RDF serialisation and thus can be said to support its reification mechanism, reified RDF statements cannot be used in OWL axioms without violating the expressiveness constraints of OWL DL.

Given these considerations, the following sections outline a design pattern that allows us to infer the summary of a reified relation. First, the general principle is described in terms of the construction of social roles. This is then elaborated to show its application for n-ary relations and other subjective constructs. Note that the purpose of this exercise is not to specify contextual subsumption along the counts-as relation (cf. Grossi (2007)) in DL, but rather to allow maximum usability of ontologically concise representations.[13]

#### Step 1: Initial Class Definition

Consider the definition of a simple social rule, a student is a person who is enrolled as such at some university. Using Searle's counts-as rule, we can rephrase this as:

> A *person* (X) counts as a *student* (Y) if enrolled at some *university* (C)

Casting this into OWL axioms, we get:

$$\begin{aligned} \mathsf{Student} \quad &\equiv \quad \mathsf{played\_by}\ \textbf{some}\ \mathsf{Person} \sqcap \\ &\qquad\ \ \mathsf{context}\ \textbf{some}\ \mathsf{University} \end{aligned}$$

#### Step 2: Constrain and Disambiguate Role Fillers

The above definition does not constrain the number of and type of entities that are valid values for the context and counts_as properties. We import the Subjective_Entity class from the LKIF Core ontology to import its restrictions. A subjective entity is defined as an entity imposed on another entity using a

[13] Section 7.5.1 describes how a role attribution can be used to 'backfire' certain property values to the brute fact.

counts-as rule:[14]

$$\begin{aligned} \mathsf{Subjective\_Entity} &\equiv \mathsf{imposed\_on}\ \textbf{some}\ \mathsf{owl{:}Thing} \sqcap \\ &\quad\ \mathsf{context}\ \textbf{some}\ \mathsf{owl{:}Thing} \\ &\sqsubseteq \mathsf{imposed\_on}\ \textbf{exactly}\ 1\ \mathsf{owl{:}Thing} \sqcap \\ &\quad\ \mathsf{context}\ \textbf{exactly}\ 1\ \mathsf{owl{:}Thing} \end{aligned}$$

$$\mathsf{imposed\_on} \equiv \mathsf{counts\_as}^{-}$$

Note the way in which the equivalent class and subclass of axioms on Subjective_Entity are used to ensure that any entity that is attributed in some context is both recognised as a Subjective_Entity, but also enforced to be attributed to exactly one entity in a single context (Section 7.2.1). We reuse the definition of Social_Role in a similar fashion. A social role is a role that can only be played by a single agent, in a single context, it inherits a cardinality restriction on the played_by and context properties from Subjective_Entity. Where played_by is defined as the inverse of a plays subproperty of counts_as. The relevant axioms from LKIF Core are:

$$\begin{aligned} \mathsf{Role} &\equiv \mathsf{played\_by}\ \textbf{some}\ \mathsf{owl{:}Thing} \\ &\sqsubseteq \mathsf{Mental\_Entity} \\ \mathsf{Social\_Role} &\sqsubseteq \mathsf{Role} \sqcap \\ &\quad\ \mathsf{played\_by}\ \textbf{some}\ \mathsf{Agent} \end{aligned}$$

$$\begin{aligned} \mathsf{played\_by} &\equiv \mathsf{plays}^{-} \\ \mathsf{played\_by} &\sqsubseteq \mathsf{imposed\_on} \\ \mathsf{plays} &\sqsubseteq \mathsf{counts\_as} \end{aligned}$$

If we adopt these axioms, the Student role can only be played by a single person at a single university. It should furthermore be clear that the redefined definition of Student is a conservative extension of the LKIF ontology:

$$\mathsf{Student} \sqsubseteq \mathsf{Social\_Role}$$

At this point, the same problem as with our definition of Transaction in Section 7.2.1 arises. Currently, we have said nothing about how the classes Person and University are related. The cardinality restrictions on the played_by and context properties allow us to infer that, since Student is a subclass of Social_Role, the Person class must be subclass of Agent. The current representation is silent as to whether a University is an agent as well, and a university could in principle be a student in the context of itself.[15] We can use our knowledge of the domain to prevent this from happening and disambiguate universities from persons, i.e. by making both classes disjoint. However, this disambiguation between context and brute fact is a *global* restriction, and indeed the properties imposed_on and context are defined as disjoint in the LKIF ontology.

[14] For a more restrictive definition we could define imposed_on and context as functional properties.

[15] In fact, a University is itself a Role played by some Organisation (which is an Agent).

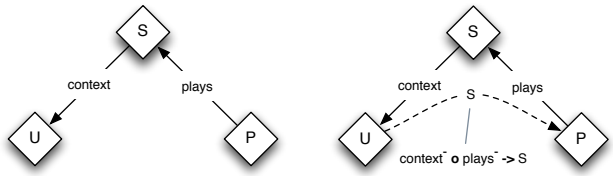


Figure 7.8: Tree-model as described by the definition of Student

#### Step 3: Traverse the Tree

Figure 7.8 shows the completion graph for the definition of Student, where the student role $S$ is played by a person $P$ within the university context $U$. As discussed earlier, this structure constitutes a reified relation expressing studentship between the university and the person. A straightforward summarisation of the reification can be specified as simple role inclusion axioms that simulate the respective named places:

$$\begin{aligned} \mathsf{context}^{-} \circ \mathsf{plays}^{-} &\sqsubseteq \mathsf{student} \\ \mathsf{plays} \circ \mathsf{context} &\sqsubseteq \mathsf{university} \end{aligned}$$

That is for the structure in Figure 7.8, a DL reasoner will infer a student relation between university $U$ and person $P$, and its inverse. Because in role inclusion axioms the property chain is a sub property of the property, the summarisation is subject to a similar limitation as RDF reification, but in the exact opposite direction. Where in summarisation we can only infer the relation given the reified structure; RDF reification only infers the reified structure, given the relation. Also, summarisation cannot be used to express an explicit predicate relation between the summarised class (Student) and the relation (student).

#### Step 4: Introduce Domain Dependence

For the purposes of our example, this role inclusion axiom is overly ambitious as it will infer the relation for *any* context and brute fact connected through a Role. The way out of this predicament is to define the student relation in terms of some of the elements particular to the structure of the Student class, i.e. to introduce domain dependence. There are two ways to do this: either by refining the properties involved, or by using **self** restrictions.

We can refine the properties in the Student definition by adding the following axioms:

$$\begin{aligned} \mathsf{Student} &\sqsubseteq \mathsf{student\_context}\ \textbf{some}\ \mathsf{University} \sqcap \\ &\quad\ \mathsf{student\_played\_by}\ \textbf{some}\ \mathsf{Person} \\ \\ \mathsf{student\_context} &\sqsubseteq \mathsf{context} \\ \mathsf{student\_played\_by} &\sqsubseteq \mathsf{played\_by} \end{aligned}$$

Because of the cardinality restrictions on the context and imposed_on properties, the properties student_context and student_played_by will correspond to

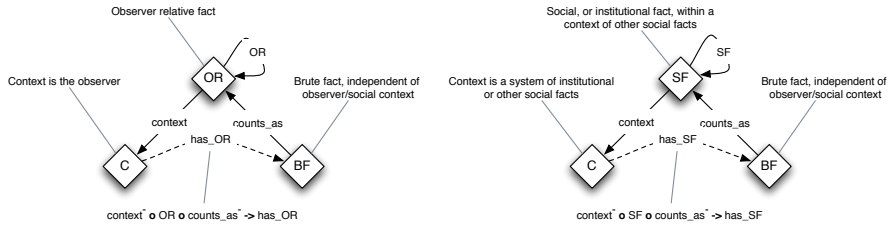


Figure 7.9: Observer-relative and Institutional facts

the context and played_by properties for any individual of type Student. Since the restriction on the domain dependent properties is only specified in a subclass axiom, inferring membership of the Student class does not depend on these properties. The student property can now be restated as:

$$\text{student\_context}^{-} \textbf{ o } \text{student\_played\_by} \sqsubseteq \text{student}$$

The main drawback of this approach is that for every type of Role, we need to add a domain dependent property and axiom for each of the properties in its definition. Furthermore, these domain dependent properties can only be used local to the Role being defined. Since both properties cannot be used to infer class membership of Student, we can infer the student relation for any three individuals connected in the prescribed manner. Using equivalence instead of subsumption in the definition of Student does not help matters, as that may make the domain *in*dependent and dependent restrictions of the Student class equivalent.

A more economical approach is to reduce the opaqueness of the distinction between the TBox and RBox by introducing a single property that uniquely identifies a class in the RBox. For our Student role, we introduce the is_student property, that will relate any student individual to itself. We can ensure this by specifying a **self** restriction on the Student class:

$$\text{Student} \sqsubseteq \text{is\_student } \textbf{some self}$$

Any individual that is related to itself via the is_student property will be identified as an instance of Student, and any individual asserted as instance of Student will be related to itself via that property. We can now rephrase the student summarisation axiom as follows:

$$\text{context}^{-} \textbf{ o } \text{is\_student } \textbf{o } \text{played\_by} \sqsubseteq \text{student}$$

The university summarisation property can be expressed by reversing the direction of the context and played_by properties. Figure 7.9 shows the general structure of this pattern for observer relative and institutional facts. This approach suffers from the same restriction regarding the equivalence axiom. Generally, the attribution of social constructs is difficult to define independently. A possible way out is to give a more specific definition of the context of the social construct, e.g. the set of actions typically associated with students. Nonetheless, the use of a *marker property* such as is_student is consistent with the fact that social facts are created via conscious acts, such as statements and declarations (Searle, 1995).

$$
\begin{aligned}
\text{Subjective\_Entity} &\equiv \text{imposed\_on } \textbf{some } \text{owl:Thing} \sqcap \\
&\quad\ \ \text{context } \textbf{some } \text{owl:Thing} \\
&\sqsubseteq \text{imposed\_on } \textbf{exactly } 1 \text{ owl:Thing} \sqcap \\
&\quad\ \ \text{context } \textbf{exactly } 1 \text{ owl:Thing} \\
\text{Role} &\equiv \text{played\_by } \textbf{some } \text{owl:Thing} \\
&\sqsubseteq \text{Mental\_Entity} \\
\text{Social\_Role} &\sqsubseteq \text{Role} \sqcap \\
&\quad\ \ \text{played\_by } \textbf{some } \text{Agent} \\
\text{Student} &\equiv \text{played\_by } \textbf{some } \text{Person} \sqcap \\
&\quad\ \ \text{context } \textbf{some } \text{University} \\
&\sqsubseteq \text{Social\_Role} \\
&\sqsubseteq \text{is\_student } \textbf{some self}
\end{aligned}
$$

$$
\begin{aligned}
\text{imposed\_on} &\equiv \text{counts\_as}^- \\
\text{played\_by} &\equiv \text{plays}^- \\
\text{played\_by} &\sqsubseteq \text{imposed\_on} \\
\text{plays} &\sqsubseteq \text{counts\_as}
\end{aligned}
$$

$$
\begin{aligned}
\text{context}^- \ \textbf{o}\ \text{is\_student}\ \textbf{o}\ \text{played\_by} &\sqsubseteq \text{Student} \\
\text{plays}\ \textbf{o}\ \text{is\_student}\ \textbf{o}\ \text{context} &\sqsubseteq \text{University}
\end{aligned}
$$

Figure 7.10: Axiomatisation of the Student class

#### Step 5: Punning

To complete the picture of summarisation, the property and class names of the subjective entity being defined can be made *the same*. OWL 2 punning allows us to treat the named entity as either an object or a class depending on how it is used (see Section 3.5.2). This way, roles such as Student can be used both as property and as class without compromising ontological commitment, or burdening (re)users of the ontology with overly verbose vocabulary. The resulting axiomatisation is depicted in Figure 7.10.

### 7.3.4 Discussion and Examples

The preceding section gives an overview of a general structure design pattern for representing the summarisation of reified relations. This summarisation is established in five steps, that partially coincide with the steps in Section 7.2.1. Create *initial class definitions* that provide a basic means for identifying individuals. Constrain the *number and type of role fillers* by applying the techniques from Section 7.2.1. Define the summarisation property as a property chain that *traverses the tree* from context to brute fact. Introduce *domain dependence* by incorporating contextual information, most notably using a *marker property* that uniquely identifies members of a class in the RBox. Optionally use *punning* to lift the syntactic distinction between the summarisation property and the corresponding reification.

The following sections give an overview of how this pattern can be applied

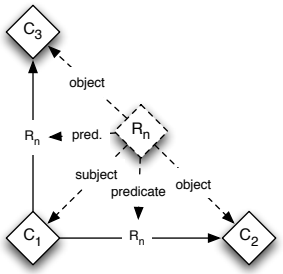


Figure 7.11: Reification $\mathsf{R}_n$ of an n-ary relation.

to various different contexts, such as the representation of n-ary relations of Section 7.3.1, and various mental entities in the LKIF Core ontology.

#### Representing N-Ary Relations

Although we have seen in Section 7.3.1 that n-ary relations cannot be represented directly in OWL DL, we can summarise them in a fashion similar to how we represent social roles. The main difference between the reification of binary relations and n-ary relations is that the latter have multiple subjects or objects, but share a single predicate. Compare the reified relation $\mathsf{R}_n$ of Figure 7.11 with that of R in Figure 7.7.

As Noy and Rector (2006) point out, n-ary relations typically have multiple objects, but a single subject (Cases 1 and 2 from Section 7.3.1). Consider their representation of the Diagnosis_Relation:

$$\begin{aligned} \mathsf{Diagnosis\_Relation} &\sqsubseteq \mathsf{diagnosis\_value}\ \textbf{some}\ \mathsf{Disease} \\ &\sqsubseteq \mathsf{diagnosis\_prob}\ \textbf{some}\ \mathsf{Probability} \\ \mathsf{Person} &\sqsubseteq \mathsf{has\_diagnosis}\ \textbf{only}\ \mathsf{Diagnosis\_Relation} \end{aligned}$$

Because diagnosis_value and diagnosis_probability are functional properties, no cardinality restrictions are defined. Figure 7.12 shows an instance of this relation $DR$ as it is related to a person $P$, a disease $D$ and its probability $P$. To make the representation more uniform, we add a restriction on the inverse of has_diagnosis as subclass axiom to Diagnosis_Relation (no diagnosis without a patient):

$$\mathsf{Diagnosis\_Relation} \sqsubseteq \mathsf{patient}\ \textbf{some}\ \mathsf{Person}$$

Following the pattern described in the previous section, we add a marker property is_diagnosis to the definition of Diagnosis_Relation:

$$\mathsf{Diagnosis\_Relation} \sqsubseteq \mathsf{is\_diagnosis}\ \textbf{some self}$$

Given these definitions, we can define the summarisation property diagnosis with domain Person and range Disease $\sqcup$ Probability as the following property inclusion axiom:

$$\begin{aligned} \mathsf{has\_diagnosis}\ \textbf{o}\ \mathsf{is\_diagnosis}\ \textbf{o}\ \mathsf{diagnosis\_value} &\sqsubseteq \mathsf{diagnosis} \\ \mathsf{has\_diagnosis}\ \textbf{o}\ \mathsf{is\_diagnosis}\ \textbf{o}\ \mathsf{diagnosis\_prob} &\sqsubseteq \mathsf{diagnosis} \end{aligned}$$

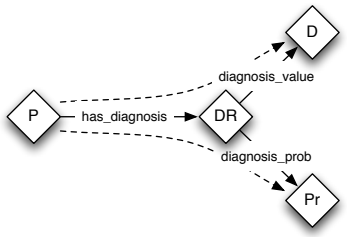


Figure 7.12: The n-ary relation $DR$ between person $P$, disease $D$ and probability $Pr$.

In this case, the marker property is_diagnosis may be a bit superfluous. However we can easily extend this pattern to express more specific kinds of diagnosis:

$$\begin{aligned} \text{Cancer\_Diagnosis} &\equiv \text{diagnosis\_value } \textbf{some } \text{Cancer} \\ &\sqsubseteq \text{is\_cancer\_diagnosis } \textbf{some self} \end{aligned}$$

and informative properties such as a has_cancer_prob property that relates the patient to the probability of such a particular diagnosis:

$$\text{has\_diagnosis } \textbf{o } \text{is\_cancer\_diagnosis } \textbf{o } \text{diagnosis\_prob} \sqsubseteq \text{has\_cancer\_prob}$$

These axioms allow us to infer for e.g. $Christine$ ($P$) that if she has a has_diagnosis relation with a Cancer_Diagnosis $DR$ which has a diagnosis_prob with value $high$ ($Pr$), then $Christine$ has_cancer_prob $high$.

This representation can be extended in a straightforward manner to cover more complex n-ary relations, as e.g. case 3 of Section 7.3.1 where $John$ buys a $Lenny\_the\_Lion$ book. His Purchase is in fact a type of Transaction (Section 7.2.1). We can use the familiar role inclusion axiom and marker property is_purchase to express relations such as buys, sells and earns:[16]

$$\begin{aligned} \text{actor}_m^- \textbf{ o } \text{part}^- \textbf{ o } \text{is\_purchase } \textbf{o } \text{part } \textbf{o } \text{object}_g &\sqsubseteq \text{buys} \\ \text{actor}_g^- \textbf{ o } \text{part}^- \textbf{ o } \text{is\_purchase } \textbf{o } \text{part } \textbf{o } \text{object}_g &\sqsubseteq \text{sells} \\ \text{actor}_g^- \textbf{ o } \text{part}^- \textbf{ o } \text{is\_purchase } \textbf{o } \text{part } \textbf{o } \text{object}_m &\sqsubseteq \text{earns} \end{aligned}$$

Figure 7.13 shows how $John$ ($Ag_1$) can be inferred to have a buys relation with the $Lenny\_the\_Lion$ book ($G$).

#### Representing Mental Entities

Besides roles, other subjective and mental entities play an important role in the LKIF Core ontology. Propositional attitudes, such as beliefs, intentions and convictions are central to the representation (and resolution) of legal cases, and in particular *mens rea*. For instance, the state of mind 'malice aforethought'

[16] Provided that the objects of both Transfer actions are suitably distinguished.

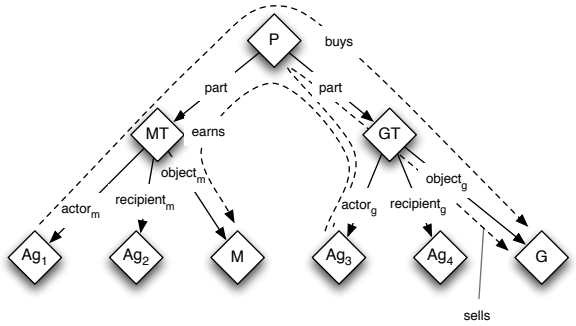


Figure 7.13: The sells, buys and earns relations in a Purchase transaction

necessary for the qualification of murder, is defined as the *intent* to kill, inflict serious bodily harm or a state of reckless indifference. Also, persons can be held liable for the actions of another person if they caused that person to hold a belief that led to an illicit act (Hart and Honoré, 1985).

Propositional attitudes are not only subjective entities – they are observer relative – but have a relational character that is very similar to that of roles. For instance, consider the following sentence:

"Mary believes that John killed Susan."

Here the propositional *content* of the belief is "John killed Susan". The pronoun 'that' is used to mark a reification of the propositional content of the belief. However, we cannot say that "John killed Susan" *is* the belief, rather the belief is inclusive of both the attitude and the content, i.e. it is "*that* John killed Suzan". We can rephrase the example as "Mary *holds* a belief *towards* the proposition 'John killed Suzan'.", as a consequence of which "Mary" *believes* "John killed Suzan".

The LKIF Core takes this duality into account and represents the classes Propositional_Attitude and Proposition as follows:

$$
\begin{aligned}
\text{Propositional\_Attitude} &\equiv \text{towards } \textbf{some } \text{Proposition} \\
&\sqsubseteq \text{Subjective\_Entity} \sqcap \text{Mental\_Object} \sqcap \\
&\quad \text{held\_by } \textbf{some } \text{Agent} \\
\text{Proposition} &\sqsubseteq \text{Mental\_Object} \sqcap \\
&\quad \text{attitude } \textbf{some } \text{Propositional\_Attitude}
\end{aligned}
$$

$$
\begin{aligned}
\text{towards} &\sqsubseteq \text{imposed\_on} \\
\text{attitude} &\sqsubseteq \text{counts\_as} \\
\text{attitude} &\equiv \text{towards}^{-} \\
\text{holds} &\sqsubseteq \text{context}^{-} \\
\text{held\_by} &\equiv \text{holds}^{-}
\end{aligned}
$$

A propositional attitude is anything held by an Agent towards some Proposition. The properties towards and holds are sub properties of the properties we used to

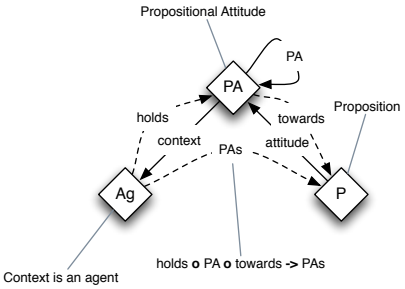


Figure 7.14: The propositional attitude $PA$ held by an agent $Ag$ towards a proposition $P$.

construct subjective entities. This definition is easily refined to cover the belief example, by adding a subclass Belief, its corresponding marker property, and the summarisation property chain:

$$\begin{aligned} \text{Belief} &\sqsubseteq \text{is\_belief } \textbf{some self} \\ &\sqsubseteq \text{Propositional\_Attitude} \end{aligned}$$

$$\text{holds } \textbf{o} \text{ is\_belief } \textbf{o} \text{ towards} \sqsubseteq \text{believes}$$

Using this definition, we can infer for any agent $Ag$ that holds a Belief ($PA$) towards a Proposition ($P$), that $Ag$ believes $P$. The general pattern is depicted in Figure 7.14).

Some attitudes are not merely internal to some agent, but are externalised by means of a speech act. The Speech_Act class in LKIF Core is defined as an action that creates a Communicated_Attitude such as declarations, assertions and promises. Communicated attitudes are those attitudes that convey an Expression, a proposition that is mediated through some Medium (e.g. a book):

$$\begin{aligned} \text{Communicated\_Attitude} &\equiv \text{states } \textbf{some} \text{ Expression} \\ &\sqsubseteq \text{Propositional\_Attitude} \\ \text{Speech\_Act} &\equiv \text{creates } \textbf{some} \text{ Communicated\_Attitude} \\ &\sqsubseteq \text{Creation} \end{aligned}$$

$$\begin{aligned} \text{states} &\sqsubseteq \text{imposed\_on} \\ \text{creates} &\sqsubseteq \text{object} \end{aligned}$$

As Creation is a subclass of Action, the Speech_Act inherits its restriction on the participants actor and result. An Agent is said to utter a Communicated_Attitude if it is created through some Speech_Act:

$$\text{Speech\_Act} \sqsubseteq \text{is\_speech\_act } \textbf{some self}$$

$$\text{actor}^{-} \textbf{ o } \text{is\_speech\_act } \textbf{o} \text{ creates} \sqsubseteq \text{utters}$$

The attentive reader may have noticed that the utters property is not defined as a sub property of the inverse of context. The reason is that if this relation were

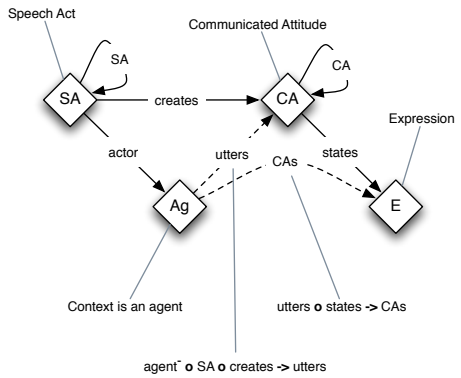


Figure 7.15: The communicated attitude $CA$ that states expression $E$, created by speech act $SA$ and uttered by agent $Ag$

asserted, the context property would become a *complex* property, and we would lose the ability to constrain its cardinality (see Section 3.5.1). We can circumvent this limitation by *wrapping* the context property into the definition of utters. This done by additionally defining the utters relation as a super property of the concatenation of context$^-$ and a marker property is_communicated_attitude:

$$\text{context}^- \circ \text{is\_communicated\_attitude} \sqsubseteq \text{utters}$$

This way, we can infer the utters relation in two ways: via the specification of a speech act, or by the explicit assertion of a communicated attitude. Unfortunately, the tree model property of DL again does not allow us to enforce that simultaneous application of both methods results in the inference that the speech acts are uttered by a *single* Agent. The same_id_as relation of Section 7.2.1 can be extended to simulate an owl:sameAs relation for this pattern:

$$\text{actor}^- \circ \text{is\_speech\_act} \circ \text{creates} \circ \text{utters}^- \sqsubseteq \text{same\_id\_as}$$

The new vocabulary can be used to express the relational character of various communicated attitudes, such as declarations and statements. For instance, the Declaration class is defined as follows:

$$\begin{aligned} \text{Declaration} &\sqsubseteq \text{is\_declaration } \textbf{some self} \\ &\sqsubseteq \text{Communicated\_Attitude} \end{aligned}$$

The marker property is_declaration is then used to define the declares relation (Figure 7.15):

$$\text{utters} \circ \text{is\_declaration} \circ \text{states} \sqsubseteq \text{declares}$$

#### Implications for Reuse

In the discussion of Chapter 6 it was hinted at that summarisation could contribute to a more practical reusability of ontologies. The broad range of use

cases for the summarisation pattern discussed in this section clearly shows that its applicability is far reaching. Without summarisation, the choice between a *relation* or *class* oriented representation of e.g. roles is an important ontological commitment. And more importantly, this choice would have to be made individually, for any of the use cases we discussed. The result is often a hodgepodge of relation and class oriented solutions, within a single ontology.

Reuse of the LKIF Core ontology in applications that use expressive features of OWL 2, such as the versioning mechanism of Klarman et al. (2008) and normative assessment described in Hoekstra et al. (2008), is quite demanding because of the large number of complex class axioms in the LKIF ontology itself (see Section 6.4). Summarisation allows us to combine the conciseness of relations with the verbose ontological correctness of classes within an ontology. A lightweight version of the ontology can be constructed by creating direct class axioms for summary properties, while discarding the axioms related to their reification. The resulting ontology is much more succinct, both qua ontological commitment and qua expressiveness. While summary properties are complex in the original ontology, the lightweight ontology represents them as simple. The latter consequently does not involve any cardinality constraints nor does it assert expressive property types over these properties. Of course ontologies that reuse the lightweight ontology may add such restrictions on those properties without violating any of the global restrictions in Motik et al. (2009), but this would not constitute safe reuse.

## 7.4 Sequences: *Change and Causation*

> "You see there is only one constant. One universal. It is the only real truth. Causality. Action, reaction. Cause and effect. "
> "And this is the nature of the universe. We struggle against it, we fight to deny it but it is of course. Pretend it is a lie, beneath our poised appearance the truth is we are completely out of control. Causality, there is no escape from it, we are forever slaves to it. Our only hope, our only peace is to understand it, to understand the why. "
>
> *Merovingian, The Matrix Reloaded*

The representation of sequences of entities is central to many domains. Examples are the representation of physically or conceptually linked structures such as chains, trains, roads, and routes, or chapters in a book, respectively (Drummond et al., 2006). For instance, Noy and Rector (2006) describe the a case for an n-ary relation that involves a route: "United Airlines flight 3177 visits the following airports: LAX, DFW, and JFK." The flight has a relation with three airports, and the sequence indicates the order in which its visits these airports. Similarly, sequences play an important role in the qualitative representation of quantities, such as relative speed or temperature. However, it is particularly essential in the representation of anything related to *time* and *change*, for clearly, temporal relations have a directed sequential structure:

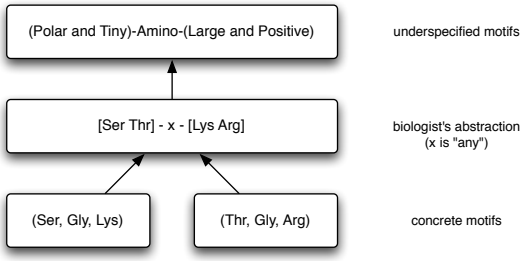


Figure 7.16: Alternative classifications of under specified protein sequence motifs (Drummond et al., 2006, Fig.2)

> Time is the conscious experiential product of the processes that allow the (human) organism to adaptively organise itself so that its behaviour remains tuned to the sequential (i.e. order) relations in its environment.
>
> (Michon, 1990, p.40)

A perceived temporal ordering of events is thus a by-product of our interaction with the order relations in reality. In Breuker and Hoekstra (2004b); Hoekstra and Breuker (2007) we have argued that this order is dependent on how processes in reality *causally* interact. The fact that the occurrence of some process is a necessary and sufficient condition for the occurrence of another process, implies a causal relation that coincides with a temporal ordering of the processes: causation is what makes time tick.

Other than the transitivity of properties, OWL does not provide a built-in construct for representing (temporal) sequences. Though OWL 2 will include an owl:datetime datatype property for representing times, a similar datatype-oriented approach is not always applicable. For instance, Drummond et al. (2006); Villanueva-Rosales and Dumontier (2007) focus on the representation of sequences of molecular structures such as protein sequences and functional groups, respectively. Proteins are sequences of amino acids, and are categorised and referred to according to properties of the patterns of amino acids – or *motifs* – they contain. Drummond et al. present a design pattern for sequences inspired by the data structure commonly used to represent *lists* (as e.g. in rdf:List, Prolog and Lisp list, and the abstract List class in Java). Their OWL-List class functions as a built-in construct, and can be used to classify proteins according to the patterns they contain (see Section 7.4.3 for a discussion of its definition). Figure 7.16 shows an example classification task in biochemistry.

The OWL representation of Villanueva-Rosales and Dumontier has a similar goal. The chemical properties of large carbon based molecules (chemical compounds) are described by experts in terms of the interaction between functional groups: partial molecular structures. Although we could envision a mapping from functional groups and motifs to unique numbers (or even a custom datatype), this moves all semantics outside of OWL itself and we can no longer adequately represent the groups themselves. Arguably, a representation of time in terms of points or intervals has a more straightforward mapping to a datatype, but again, such an approach does not define these temporal occurrences themselves.

The following sections introduce a design pattern for representing sequences, based on a use case for causal explanation. The next section introduces the perspective on causation of Hoekstra and Breuker (2007) and illustrates requirements for its representation as part of LKIF Core. Section 7.4.3 compares the pattern with that of Drummond et al. and discusses its advantages and limitations.

### 7.4.1 Causation

Causality plays a central role in virtually all scientific disciplines. In all cases its use is more precise than in commonsense – for instance, it is distinguished from covariance, correlation and coincidence – but essentially it does not differ in its capacity to explain dependencies between events. Lehmann (2003) adopts a distinction between the notions of causality and *causation*; he uses the term 'causality' to denote the ontological view, and reserves the term 'causation' for the *occurrence* of causality. The definition of causality is a major issue in philosophical metaphysics, and has been for many centuries (Kim, 1998; Davidson, 2001; Schaffer, 2003). Philosophy primarily concerns itself with what causality *is*, i.e. it is focused on ontological questions regarding the existence and properties of causality. Epistemological questions arise concerning how we can know about the occurrence of causality in the real world and, once we do know, what inferences we can draw from the causal relation between two events.

Causation is an abstract, reflective concept that summarises a more qualitatively distinct relation. An account of causation involves an understanding of relationships between events in terms of *processes*. This intermediary role of processes can be conceived of as a dependency: as causation between events. Causation is no more and no less than the (abductive) inference that the occurrence of event $B$ can be explained by a (sub-)process that is implied by an earlier or simultaneous event $A$. In other words, after we perceive a collision between two billiard balls, we infer a transfer of force from the moving ball to the static ball. This allows us to say that the collision caused the second ball to move. In a nutshell, recognising the transfer of power at the collision is sufficient for our understanding; the assertion that the collision caused the ball to move is superfluous.

This view furthermore exposes the *overloading* of causal relations common in many domains, such as the interspersion of liability and causality in legal causation (Hoekstra and Breuker, 2007). In general, we can distinguish two types of causation: *physical causation* that describes physical processes and *agent causation*, which describes the actions of rational agents. Legal theory introduces two additional forms of causation (Lehmann, 2003; Hart and Honoré, 1985): *interpersonal causation*, which describes the effect one agent might have on another (e.g. as a consequence of communication), and *negative causation*, representing the connection between an effect and some agent *not* acting (e.g. negligence). As discussed in Hoekstra and Breuker (2007), the latter are legal constructs rather than basic causal relations.

### Approaches

In Artificial Intelligence, causation has been the direct and indirect object of study in a number of specialised areas. Worth mentioning in this respect is the work of Pearl (2000), which finds its basis in a *probabilistic* representation of physical causation; more knowledge-intensive approaches to *physical causation* from the fields of Qualitative Reasoning (QR) and Model Based Reasoning (MBR) (e.g. Horn (1990)), *ontology-based* approaches, and a multitude of formalisms for *agent causation* (communication) within multi-agent systems.

**Formal Causal Reasoning** The work by Pearl (2000) stands out in AI as the most comprehensive and explicit modelling of what could be called "computational causality". His main contribution lies with probabilistic (Bayesian) modelling of quantifiable dependencies between occurrences, e.g. the use of drugs and their effects on patients. In this sense it fits the tradition in science to enable the identification of causes or causal factors in a well founded, formal way. This work is also relevant for legal reasoning about cases that have a probabilistic basis, as for instance in claims about compensation for damage of health suffered due to the consumption of certain industrial products. However, his approach is not particularly well suited for a more qualitative rather than quantitative conception of causation, which Pearl (2000, Ch.10) coined *actual cause*:

> "an event recognised as responsible for the production of a certain outcome [...] Human intuition is extremely keen in detecting and ascertaining this type of causation and hence is considered the key to construct explanations [...] and the ultimate criterion (known as "cause in fact") for determining legal responsibility"
> [...]
> "Clearly, actual causation requires information beyond that of necessity and sufficiency: the actual process mediating between the cause and the effect must enter into consideration."
>
> Pearl (2000, p.309)

However, Pearl sees the conceptual basis for actual causation as a mere additional element, rather than primary and sufficient for deciding on causation in fact. His somewhat exploratory work on factual causation is aimed at a formal, 'correct' modelling of causal dependency rather than a more qualitative commonsense perspective.

**Qualitative and Model Based Reasoning** Another potential source of inspiration for the representation of causal relations is the representation and reasoning involved in modelling the structure and behaviour of systems (Bredeweg and Struss, 2004). Qualitative reasoning (QR) started as an approach to commonsense reasoning and initially coined *naive physics*. In his influential second naive physics manifesto, Hayes (1985) argues that causality is not a "useful, self-contained theory", but "that it is an umbrella term for a large variety of particular relationships" (Hayes, 1985, p.19). In Hoekstra and Breuker (2007) we underwrite his stance that causality is not a primary term in an ontology

of common sense, e.g. it is not defined by LKIF Core. Rather, the variety of causal relationships between events considered to correspond to the relations between different kinds of processes.

In QR, a model of the structure of a system is used to simulate the propagation of changes through the system, including structural changes of the system itself. This propagation yields chains of events, which represent a prediction of the behaviour of the system, given some initial state. Events in QR are connected by property values induced by two types of relations: *influences* and *proportionalities*. Proportionalities represent definitional or inherent dependencies between property values. For instance, an increase in volume of a substance has a linear correspondence to its weight.[17] On the other hand, influences represent an actual *causal* effect on the value of properties, for instance the proximity of a heat source will influence the temperature of an object. These causal relations follow from processes (Forbus, 1984), which are represented as *model fragments*.

QR is particularly useful for modelling well known and stable physical structures. For instance, model based reasoning – an applied branch of QR – is used mainly in the representation and diagnosis of devices. Although QR techniques have been used in other domains than physics, e.g. in modelling ecological and social systems (Bredeweg et al., 2006), it is limited to purely *physical causation* and does not provide a means to model agents, let alone the processes initiated by them. Furthermore, as discussed in Hoekstra et al. (2006), the model fragments that play a central role in QR cannot be represented in OWL in a straightforward manner, as they are highly structured concepts (Motik et al., 2007a, and Section 7.1).

**Agent Technology** Agent causation is not explained just by causes but by reasons as well, and in particular the *intention* to perform an action. The notion of intention, although more recent, has given rise to as much philosophical controversy as causation. In AI the notion has been operationalised to model actions, and in particular communication between artificial agents. The languages by which these agents communicate typically provide constructs for representing belief-states, actions, plans, data etc. The most prominent example being the belief, desire, intention model of Georgeff et al. (1999, BDI). The BDI model is a representation of the inner workings of agents (including people), and is intended to contribute to a better understanding of how intention influences action. Nonetheless, there are a number of reasons why agent technology cannot be used for detecting commonsense causation. First of all, agent technology is concerned with *simulation*, the realtime behaviour of agents, where our use case relies on post-diction. Furthermore, the BDI model does not cover physical causation.

**Ontology** The ontological approach to causation in fact described in Lehmann et al. (2004); Lehmann and Gangemi (2007), defines causal dependencies within the framework of the DOLCE ontology, see e.g. Gangemi et al. (2002, and Section 6.2.1).

Causal relations relate very simple events that change a single aspect of a single object. Between these events, three existential dependencies are identi-

[17] Provided that the substance is not a contained gas.

fied: *structural*, *causality* and *circumstantial* dependencies. Physical causation is defined as the relation that holds between two individual events that satisfy both the causality and the circumstantial constraints. The framework allows for the classification of some description of a number of events as an instance of *physical causation in fact*.

The ontology contains no theory of the (physical) world; i.e. descriptions are not expressed using domain knowledge. For instance, the semantics of a state is not represented intensionally, but only through its name (e.g. *being-wounded*), it contains no description of *what it means* to be in that state. The lack of such descriptions makes this approach less useful for explaining the existence of a causal relation: the basic causal relations between events are *asserted* rather than inferred on the basis of domain knowledge. A system built on such a representation is less flexible in dealing with incomplete knowledge, as it can only infer causal propagation on the basis of a fully specified, explicit causal model.

### 7.4.2 Representing Causal Change

This section introduces a content pattern for describing the linear ordering of events that lies at the heart of causal relations in LKIF Core. The approach differs from Drummond et al. (2006) in that it captures the sequential nature of these relations, similar to how transitivity relates to the part of relation, whereas Drummond et al. use a nested structure. The pattern is based on the principle that an account of subsequent states of objects, expressed in terms of domain knowledge, can be recognised as the occurrence of one or more processes. The interaction between these processes is both necessary and sufficient for the identification of causal relationships (Hoekstra and Breuker, 2007). Given the sequential ordering of these states and processes we can furthermore infer temporal relations.

#### Step 1: Initial Class Definition

Events and states occur at some time and place, i.e. they happen against a four dimensional canvas of space and time (Davidson, 2001). These co-ordinates determine the possibility of causal relationships: time and location limit causal propagation. The intensional definition of Change expresses a difference between before and after its instances occur, and thus involves a requirement and a result situation. A particular situation may be any valid configuration of entities. Since these entities have to be *related* in order for them to form a single situation, it can be generically captured by an OWL class description that takes one of the entities participating in the situation as focal point. In other words, where the definition of a change at class level describes required and resulting *situations*, a particular occurrence of a change, i.e. an *event*, is related to single individuals.[18] Processes are changes that consist of other changes, though not every change that is composed of other changes is a process.

[18]States and events are *occurrences* and are used to respectively refer to individual objects and changes.

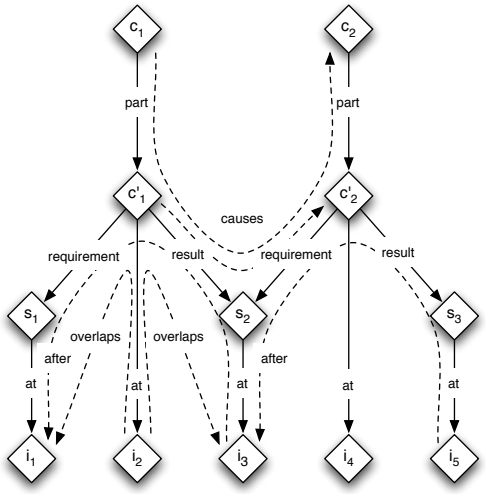


Figure 7.17: Change and causal relations.

The initial class definitions of Change and Process are thus as follows:

$$\begin{aligned}
\text{Change} \quad &\equiv \quad \text{requirement } \textbf{some } \text{owl:Thing} \sqcap \text{result } \textbf{some } \text{owl:Thing} \\
&\sqsubseteq \quad \text{requirement } \textbf{exactly } 1 \sqcap \text{result } \textbf{exactly } 1 \\
&\sqsubseteq \quad \text{at } \textbf{some } \text{Place} \sqcap \text{at } \textbf{some } \text{Temporal_Occurrence} \\
&\sqsubseteq \quad \text{part } \textbf{only } \text{Change} \\
\\
\text{Process} \quad &\sqsubseteq \quad \text{Change}
\end{aligned}$$

### Step 2: Traverse the Tree

The above definition already allows us to define a number of interesting causal and temporal relations. For instance, consider the situation in Figure 7.17. The changes $c_1$ and $c_2$ each have a part $c_1'$ and $c_2'$ where the result $o_2$ of $c_1'$ is a requirement for $c_2'$. Here, clearly $c_1'$ causes $c_2'$, but $c_1$ can be said to cause $c_2$ as well. These causal relations are straightforwardly defined using two role inclusion axioms:

$$\begin{aligned}
\text{result } \textbf{o } \text{requirement}^- \quad &\sqsubseteq \quad \text{causes_directly} \\
\text{causes_directly} \quad &\sqsubseteq \quad \text{causes} \\
\text{part } \textbf{o } \text{causes } \textbf{o } \text{part}^- \quad &\sqsubseteq \quad \text{causes}
\end{aligned}$$

where causes is a transitive property and causes_directly is not. We can furthermore remark that if two situations are the requirement and result of a change, then a temporal ordering relation must hold between the intervals at which they hold:

$$\begin{aligned}
\text{at}^- \textbf{ o } \text{result}^- \textbf{ o } \text{requirement } \textbf{o } \text{at} \quad &\sqsubseteq \quad \text{after} \\
\text{at}^- \textbf{ o } \text{requirement}^- \textbf{ o } \text{result } \textbf{o } \text{at} \quad &\sqsubseteq \quad \text{before}
\end{aligned}$$

These inclusion axioms allow us to infer e.g. that the interval $i_1$ holds before $i_3$, and $i_3$ before $i_5$. However, for a change to affect its conditional situations, we may require its temporal occurrence to coincide with an interval $i_2$ that partially overlaps the time of its required and resulting situation:

$$\begin{aligned} \mathsf{at}^- \circ \mathsf{requirement} \circ \mathsf{at} &\sqsubseteq \mathsf{overlaps} \\ \mathsf{at}^- \circ \mathsf{result} \circ \mathsf{at} &\sqsubseteq \mathsf{overlaps} \end{aligned}$$

A more rigorous adoption of Allen (1984) might specify that $i_2$ *finishes* the interval $i_1$ and *starts* interval $i_3$. However, this would mean that $i_2$ coincides with an overlap between $i_1$ and $i_2$, which is a significant ontological commitment to the simultaneous coexistence of the situations $s_1$ and $s_2$. To exclude this possibility, the requirement and result properties are made disjoint. This also removes models where a change is inferred to directly cause itself.

We can now also specify that a Process (or any other entity) occurs during an interval that covers, i.e. encompasses, all intervals of all changes it is composed of:

$$\mathsf{at}^- \circ \mathsf{part} \circ \mathsf{at} \sqsubseteq \mathsf{covers}$$

Corresponding spatial relations, such as overlaps can be constructed in a similar fashion.

#### Step 3: Disambiguate Role Fillers

The causes property allows us to define Process as the class of causal changes:

$$\begin{aligned} \mathsf{Process} &\sqsubseteq \mathsf{Change} \sqcap \mathsf{causes}\ \mathbf{some}\ \mathsf{owl{:}Thing} \\ &\sqsubseteq \mathsf{part}\ \mathbf{only}\ (\mathsf{Change} \sqcap \mathsf{causes}\ \mathbf{some}\ \mathsf{owl{:}Thing}) \end{aligned}$$

However, this definition leaves room for changes that are causal without being (part of) a process. In fact, as argued in Hoekstra and Breuker (2007) causation does not exist independently from processes. Whether some change is causal is determined by whether it is either a process or part of a process, and not the other way around. The causes relation should therefore only hold between entities that meet these criteria. We alter the definition of Process and causes by including a marker property is_process (see Section 7.3.3):

$$\mathsf{Process} \sqsubseteq \mathsf{is\_process}\ \mathbf{some}\ \mathbf{self}$$

$$\mathsf{is\_process} \circ \mathsf{part} \circ \mathsf{causes} \circ \mathsf{part}^- \circ \mathsf{is\_process} \sqsubseteq \mathsf{causes}$$

The above property chain infers a causes relation between two processes, but it does not specify a causal relation between the changes that are part of these process. To do this, we introduce the class Causal_Change and a marker property is_causal_change that captures those changes that are part of processes. The definition of Process is altered accordingly:

$$\mathsf{part\_of} \equiv \mathsf{part}^-$$

$$\begin{aligned}
\mathsf{Causal_Change} &\equiv \mathsf{part_of}\ \textbf{some}\ \mathsf{Process}\\
&\sqsubseteq \mathsf{is_causal_change}\ \textbf{some self}\\
&\sqsubseteq \mathsf{Change}\\
\mathsf{Process} &\sqsubseteq \mathsf{is_process}\ \textbf{some self}\\
&\sqsubseteq \mathsf{Change} \sqcap \mathsf{causes}\ \textbf{some}\ \mathsf{owl{:}Thing}\\
&\sqsubseteq \mathsf{part}\ \textbf{only}\ \mathsf{Causal_Change}
\end{aligned}$$

Anything that is part of some process, is a causal change, and processes consist only of causal changes. As a side note, we can now define Cause as an Epistemic_Role played only by causal changes:

$$\mathsf{Cause} \equiv \mathsf{Epistemic\_Role} \sqcap \mathsf{played\_by}\ \textbf{some}\ \mathsf{Causal\_Change}$$

To define the causal relation between causal changes, we replace the property chain for causes relations between changes (the second role chain in Step 2) with the following role inclusion:

$$\mathsf{is\_causal\_change}\ \textbf{o}\ \mathsf{causes\_directly}\ \textbf{o}\ \mathsf{is\_causal\_change} \sqsubseteq \mathsf{causes}$$

The inverse of the causes property is defined in a straightforward manner. The transitivity of the causes relation allows us to infer causal propagation both over a chain of connected result and requirement_of properties, and between processes that are connected via such a chain. Furthermore, the current definition ensures that causal relations only hold between changes and processes at the same aggregation level. A single change cannot cause a process, but rather the process (as a whole) that contains that change does.

#### Step 4: Introduce Asymmetry

The representation presented in the previous step allows us to infer causal relations between changes and processes provided that we *know* some change constitutes a process. However, as we have also said, the recognition of a process constitutes a causal *explanation*; the causal relations follow from this recognition. We therefore need some way to qualitatively distinguish different types of processes.

To illustrate this, consider a very simple blocks world that consists of a single Block (in three states $b_1$, $b_2$ and $b_3$), and two Surface individuals ($s_1$ and $s_2$). Any Object can be positioned on a single other individual of type Object, and the only operation available is to move an object to a different position. Our Move process will consist of lifting the object from $s_1$ and putting it on $s_2$.[19]

[19]Of course this is a simplification over the complexity of even the simplest processes in reality, such as the interplay between mass, speed, forces, and potential and kinetic energy in a collision between billiard balls. However, incorporating these properties does not alter the structure presented here.

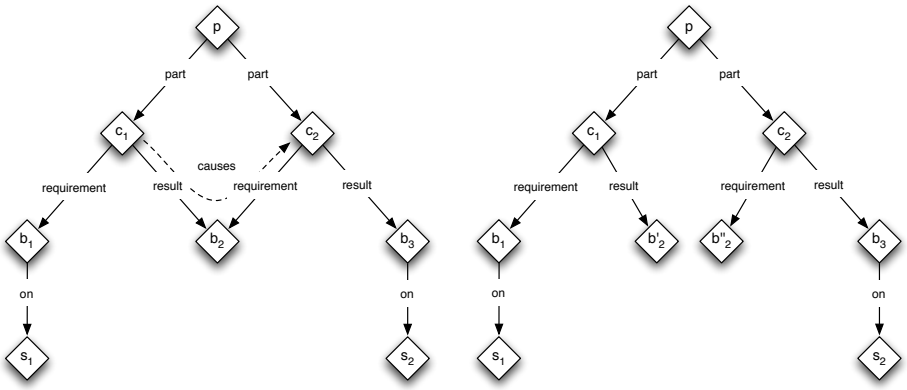


Figure 7.18: Two valid models of the class Move.

The corresponding class descriptions and individual assertions are as follows:

$$
\begin{array}{rcl}
\mathsf{Move} & \equiv & \mathsf{part}\ \textbf{some}\ \mathsf{Lift} \sqcap \mathsf{part}\ \textbf{some}\ \mathsf{Drop} \\
& \sqsubseteq & \mathsf{Process} \\
\mathsf{Lift} & \equiv & \mathsf{requirement}\ \textbf{some}\ (\mathsf{Object} \sqcap \mathsf{on}\ \textbf{some}\ \mathsf{Object}) \\
& & \sqcap\ \mathsf{result}\ \textbf{some}\ (\mathsf{Object} \sqcap \textbf{not}(\mathsf{on}\ \textbf{some}\ \mathsf{Object})) \\
\mathsf{Drop} & \equiv & \mathsf{requirement}\ \textbf{some}\ (\mathsf{Object} \sqcap \textbf{not}(\mathsf{on}\ \textbf{some}\ \mathsf{Object})) \\
& & \sqcap\ \mathsf{result}\ \textbf{some}\ (\mathsf{Object} \sqcap \mathsf{on}\ \textbf{some}\ \mathsf{Object}) \\
\mathsf{Object} & \equiv & \{b_1, b_2, b_3, s_1, s_2\} \\
& \sqsubseteq & \mathsf{on}\ \textbf{max}\ 1 \\
\mathsf{Block} & \sqsubseteq & \mathsf{Object} \\
\mathsf{Surface} & \sqsubseteq & \mathsf{Object}
\end{array}
$$

Note that for the definition of Lift and Drop we need to close the world by enumerating all possible members of the Object class. This allows us to specify that $b_2$ is not on *any* other object using negative property assertions. Without this, the class restriction **not**(on **some** Object) would never be satisfied. The process $p$ consisting of changes $c_1$ and $c_2$ on the blocks is defined as in Figure 7.18. Because $b_2$ is both the result of $c_1$ and the requirement for $c_2$, a causal relation is inferred between these two changes. However, the definition of Move has a valid model where the two changes are not connected in this way (Figure 7.18), and no causal relation is inferred.

As it is exactly the sequential character of changes that defines processes, we make the recognition of the process entirely dependent on a property chain that expresses its characteristic causal sequence of changes. This chain is (again) defined using marker properties that allow us to recognise classes inside the

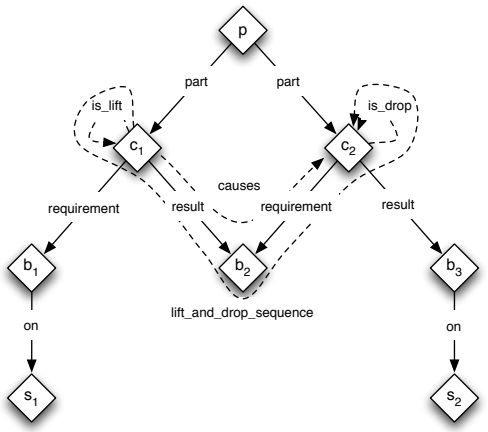


Figure 7.19: Example of a process sequence.

RBox:

$$
\begin{aligned}
\mathsf{Lift} &\sqsubseteq \mathsf{is\_lift}\ \textbf{some self}\\
\mathsf{Drop} &\sqsubseteq \mathsf{is\_drop}\ \textbf{some self}\\
\mathsf{Move} &\equiv \mathsf{part}\ \textbf{some}\ (\mathsf{lift\_and\_drop\_sequence}\ \textbf{some}\ \mathsf{owl{:}Thing})\\
&\sqsubseteq \mathsf{part}\ \textbf{some}\ \mathsf{Lift}\\
&\sqsubseteq \mathsf{part}\ \textbf{some}\ \mathsf{Drop}\\
&\sqsubseteq \mathsf{Process}
\end{aligned}
$$

$$\mathsf{is\_lift}\ \textbf{o}\ \mathsf{causes}\ \textbf{o}\ \mathsf{is\_drop} \sqsubseteq \mathsf{lift\_and\_drop\_sequence}$$

Of course, a process can be defined by any number of such sequences. However, an obvious limitation of the representation is that it does not enforce the changes in the process sequence to be a part of the process itself. Parthood can be inferred, but this requires the representation of a separate role chain for every step in the process sequence as traversing the next property would include all changes (indirectly) caused by the process.

Figure 7.19 shows how the lift_and_drop_sequence traverses the decomposition of process $p$. The marker properties allow us to distinguish between the different branches of the decomposition in the same way that we introduced asymmetry in Section 7.2.1.

### 7.4.3 Discussion

The previous section presents an approach to the representation of sequences based on four familiar steps. First an *initial class definition* gives the basic structure for the elements of the sequential structure. Secondly, a transitive property is defined by *traversing* the basic structure. Additionally, the connected nature of the structure allows us to infer more relations between its elements (e.g. the temporal relations between the occurrence of changes). The third step is to *disambiguate* role fillers by introducing marker properties that ensure the sequence

relation only holds between appropriate elements. Finally, *asymmetry* is used to expose the sequential character of the structure. We mark a starting element that defines the sequence as a composition of property chains.

Drummond et al. (2006) and Noy and Rector (2006) take a different approach, and represent sequences as *lists*. The OWLList class of Drummond et al. is defined as follows:

$$\begin{aligned} \mathsf{OWLList} &\sqsubseteq \mathsf{isFollowedBy}\ \textbf{only}\ \mathsf{OWLList} \\ \mathsf{EmptyList} &\equiv \mathsf{hasContents}\ \textbf{max}\ 0 \\ &\equiv \mathsf{OWLList} \sqcap \neg(\mathsf{isFollowedBy}\ \textbf{some}\ \mathsf{owl{:}Thing}) \\ \\ \mathsf{hasNext} &\sqsubseteq \mathsf{isFollowedBy} \end{aligned}$$

where hasContents and hasNext are functional properties, and isFollowedBy is transitive. In this definition the hasContents property has the same role as marker properties in the pattern presented of the previous section, where the hasNext property is a generalisation of the result and requirement properties, and the transitivity of isFollowedBy corresponds to the causes relation. The OWLList pattern is more verbose as for every member of the sequence, an additional OWLList individual needs to be created that links the contents to the next member. To see the difference, consider a (Ser, Gly, Lys) protein sequence from Figure 7.16 using the OWLList pattern:

$$\begin{aligned} \mathsf{Ser\text{-}Gly\text{-}Lys} \equiv\ & \mathsf{OWLList} \sqcap \mathsf{hasContents}\ \textbf{some}\ \mathsf{Ser} \sqcap \\ & \mathsf{hasNext}\ \textbf{some}\ (\mathsf{OWLList} \sqcap \mathsf{hasContents}\ \textbf{some}\ \mathsf{Gly} \sqcap \\ & \mathsf{hasNext}\ \textbf{some}\ (\mathsf{OWLList} \sqcap \mathsf{hasContents}\ \textbf{some}\ \mathsf{Lys} \sqcap \\ & \mathsf{EmptyList})) \end{aligned}$$

With some appropriate definition of the classes Ser, Gly and Lys. Using the role-based pattern, we get:[20]

$$\begin{aligned} \mathsf{Ser} &\sqsubseteq \mathsf{is\_ser}\ \textbf{some self} \\ \mathsf{Gly} &\sqsubseteq \mathsf{is\_gly}\ \textbf{some self} \\ \mathsf{Lys} &\sqsubseteq \mathsf{is\_lys}\ \textbf{some self} \\ \mathsf{Ser\text{-}Gly\text{-}Lys} &\equiv \mathsf{part}\ \textbf{some}\ (\mathsf{serglylys\_seq}\ \textbf{some}\ \mathsf{Lys}) \end{aligned}$$

$$\mathsf{is\_ser}\ \textbf{o}\ \mathsf{next}\ \textbf{o}\ \mathsf{is\_gly}\ \textbf{o}\ \mathsf{next}\ \textbf{o}\ \mathsf{is\_lys} \sqsubseteq \mathsf{serglylys\_seq}$$

Compared to Drummond et al. (2006) the approach presented here has several advantages. First of all, the pattern is much less verbose at both the class and individual level (see Figure 7.20). Furthermore, the pattern does not rely on the representation of a data structure: it feels conceptually odd to call Ser-Gly-Lys a *list*, where it is really a molecular structure composed of three substructures. This is because the OWLList reflects an epistemological, rather than an ontological perspective.

[20] For clarity, the two approaches use distinct property names. Similarly, marker properties are prefixed with 'is_' though more intuitive names are allowed through punning, e.g. Ser instead of is_ser.

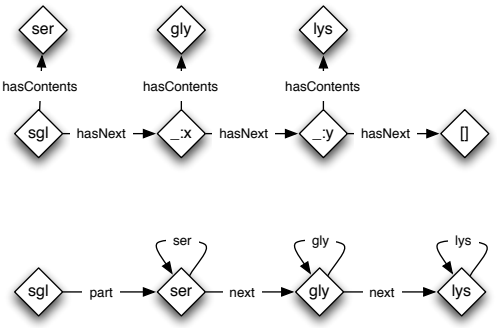


Figure 7.20: The (Ser,Gly,Lys) sequence represented as OWLList and as serglylys_seq.

One of the main problems with the OWLList class definition reported by Drummond et al. is that some of its valid models are not proper lists: it does not exclude the additional branches that can be defined using isFollowedBy instead of the functional hasNext property. Although the intended interpretation of the isFollowedBy property is that it is the transitive closure of hasNext, this cannot be enforced in OWL. The role-based approach suffers from the same problem, e.g. causes and serglylys_seq are super properties of their respective chains, but are not *equivalent* to it: an individual can be asserted to have a serglylys_seq property to some Lys individual without being connected through each consecutive step of the property chain (see Section 7.2.2). However, here the problem is less salient as the properties are specific to the class being defined (e.g. Ser-Gly-Lys), where isFollowedBy is domain independent and transitive. Similarly, a limitation shared by both approaches is that neither can exclude cycles, as this would require the complex sequence properties to be antisymmetric.

Where OWLList depends on the sequence to have no branches, and uses the functional hasNext property to (partially) ensure this, the role approach does allow for arbitrary branching, by enforcing or lifting local cardinality constraints on the next property (or corresponding alternative). For instance, the causal propagation defined in the previous section allows a state to be connected to multiple changes via the requirement property. Two process sequences that partially overlap can be superimposed by maintaining an exact cardinality restriction of 1 on the part property, while defining an existential restriction on both process sequence properties:

$$\begin{aligned} \mathsf{SomeProcess} \quad &\equiv \quad \mathsf{part}\ \mathbf{some}\ (\mathsf{process\_sequence}_1\ \mathbf{some}\ \mathsf{A} \sqcap \\ &\qquad\quad \mathsf{process\_sequence}_2\ \mathbf{some}\ \mathsf{B}) \\ &\sqsubseteq \quad \mathsf{part}\ \mathbf{exactly}\ 1 \end{aligned}$$

Alternatively, we can prohibit branching over the next property by defining next as functional, or specifying a maximum cardinality restriction on the property for each protein class.

Furthermore, sequences of a specific length are hard to define in either approach as no cardinality restrictions are allowed on transitive and other complex properties (See Section 3.5.1). The OWLList pattern requires an exhaustive

representation of each member class, while the role pattern needs the definition of a role inclusion axiom of the intended length. The advantage of the latter solution is that it can simply be stated in terms of the generic next property, whereas the former requires a rather complex class definition. The reason is that where role inclusion axioms are *sequential*, and can easily be concatenated, class descriptions are *nested* and do not allow direct reuse of generic structures.

Suppose the definition of simple sequence $S = [a, b, x, x, e]$, where $x$ is an arbitrary element. The OWLList pattern for this sequence would be:

$$S = [a, [b, [x, [x, [e, []]]]]]$$

and it can readily be seen that the representation of the sequence $[x, x]$, namely $[x, [x, []]]$ cannot be directly inserted in the pattern. On the other hand, the role pattern representation of the sequence is:

$$S = a \circ b \circ x \circ x \circ c$$

where $x \circ x$ can be substituted with $Xs = x \circ x$, resulting in $S = a \circ b \circ Xs \circ c$. In a similar fashion, OWLList is limited in the representation of sub-lists, and requires the explicit representation of the entire list. The role approach does not have this limitation.

The procedure for causal explanation described in Breuker and Hoekstra (2004b); Hoekstra and Breuker (2007) was based on the stance of e.g. Michon (1990) that the experience of time and causation are cognitive by-products of our interaction with reality. The discovery of causal relations is therefore largely a matter of 'making sense' of changes in the world around us (cf. the cognitive perspective outlined in Section 6.2.2), the experience of time is a by-product of this process. Admittedly, the representation of processes and causal propagation presented here reflects a simplified view on causation. For instance, because of the limitations of OWL, it does not cover causal relations between a change where the *non* existence of a certain object is a requirement, and a change that ensures its *non* existence (i.e. a change that does not have a result).[21]

The system presented in Breuker and Hoekstra (2004b) relied on rules and a search over the knowledge base to 'probe' for possible processes in a bag of separate individuals. A match with a known process would indicate a causal and temporal ordering. The pattern introduced in the previous section can be used to achieve the same goal, but infers the types of processes occurring in the knowledge base by using an OWL 2 DL reasoner.[22]

[21] In fact, although it is possible to assert an individual that represents an object's non-existence, we cannot prevent through DL constructs that an 'existence' and 'non-existence' individual are asserted at the same time.

[22] Given the expressiveness of the pattern, this is bound to be computationally expensive for larger sets of states.

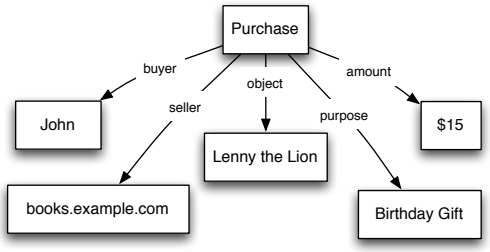


Figure 7.21: Transaction example from Noy and Rector (2006).

## 7.5 Patterns in Practice: *Action!*

The notion of *action* is a complex concept that brings together the perspectives of all three patterns defined in the preceding sections. The Transaction described in Section 7.2 involved two Transfer actions. The Purchase of Section 7.3 is such a transaction, and was conceived as a single reified n-ary relation. Its representation involved the inference of several relations between participants in the transaction. Furthermore, actions can be seen as the result of attributing an intention to some agent in starting a process (see also Section 6.3.2). In other words, an action is a Subjective_Entity that is imposed_on a Process. The connection between actions and processes is tight to such an extent, that the connection between an agent and the consequences of its actions is conceived of as *causal* (Davidson, 2001; Lehmann, 2003). In this light it is important to distinguish *reasons* from *causes*, i.e. "Sally hit Jimmy on the head *because* she was angry" gives an explanation of Sally's intention to hit Jimmy, and *not* of a causal relation between her anger and the hitting.

One might argue that since actions and processes *coincide*, action-processes cannot cease to be actions, the Action class must be rigid and should thus be defined as a subclass of Process. This would be possible if actions (and transactions) had the spider-like structure of Joe's Purchase in Noy and Rector (2006, and Section 7.3.1) (see Figure 7.21).[23] However, Breuker (1981, Ch.6) shows that the surface structure of linguistic expressions is a deceptively simplistic abstraction of a more complex conceptual structure. This conclusion is based on an analysis of mental processes that take place when people read a pronoun, such as 'it', and find its referent. The difference in response time for subjects in performing this reference assignment gives insight in the conceptual structure that exists in their mind.

Actions are often represented in terms of relationships with certain *thematic roles*, such as actor (or agent), object, instrument, location, recipient, patient, theme, etc. Over the years, many alternative representations of the relation between actions and thematic roles have been, and still are, investigated. Breuker's research corroborated the structure of conceptual dependency graphs, or *scripts*, of Schank and Abelson (1975), where characteristically the *instrument* of an action has a subordinate role. In particular, Breuker suggests that thematic roles exist at three distance levels from the action:

[23] Figure 7.13 in Section 7.3.4 already gave an alternate representation based on the Transaction pattern.

- The first level contains the *agent* and *object* roles, these are the most central to the action.
- At the second level, its *recipient* and *location* are specified. These roles can be used both in the sense of antecedent and consequent of the action, e.g. the from-location vs. the to-location.
- The *instrument* role is specified at the third level, and reflects a subordinate state or action necessary for the action to take place.

A second reason to conceive actions as subjective entities, rather than processes, is that action frames can range from very simple, as in "The butcher cuts the meat", to very complex actions such as "With a knife, the butcher cuts the meat into convenient pieces for the nice old lady." A representation of such sentences should adequately cover the structure and intentional perspective of the action expressed by it, without interfering with the structure and causal perspective of processes. Thirdly, the ability to conceive of actions without them actually taking place in reality, such as in planning, is an indication that they are mental entities. The role of verbs in the creation of metaphors (Pinker, 2007, and Section 5.6.1) furthermore suggests that the default relata of actions are *roles*. They are slots that have certain default fillers, but can be reapplied to non-standard categories to create a metaphor.

The following section shows how the three patterns are combined to create a consistent representation of the class Action.

### 7.5.1 Representing Action

As identified by Breuker (1981), the most basic participants of an Action are its actor and object. Only Agents can be the actor of an Action. Furthermore, Actions may be imposed_on a Process. We specify its basic class definition as follows:

$$\begin{aligned} \mathsf{Action} &\equiv \mathsf{actor}\ \mathbf{some}\ \mathsf{Agent} \sqcap \mathsf{object}\ \mathbf{some}\ \mathsf{owl{:}Thing} \\ &\sqsubseteq \mathsf{Subjective\_Entity} \sqcap \mathsf{actor}\ \mathbf{only}\ \mathsf{Agent} \sqcap \\ &\quad\ \mathsf{imposed\_on}\ \mathbf{max}\ 1\ \mathsf{Process} \end{aligned}$$

$$\begin{aligned} \mathsf{object} &\sqsubseteq \mathsf{participant} \\ \mathsf{actor} &\sqsubseteq \mathsf{participant} \end{aligned}$$

Note that this definition differs from the representation of Transfer in Section 7.2.1. It is less strict as it does not involve cardinality restrictions on the properties involved. Because we will *infer* the various roles of participants in the action using role inclusion axioms, we can no longer use number restrictions. In fact, as participants may reoccur at multiple stages of the process, e.g. the various states a piece of meat is in while being cut, there is a good non-technical reason for lifting the number restriction as well. However, it does mean that we need to specify additional conditions for the same_id_as property:

$$\begin{aligned} \mathsf{object}^- \circ \mathsf{object} &\sqsubseteq \mathsf{same\_id\_as} \\ \mathsf{actor}^- \circ \mathsf{actor} &\sqsubseteq \mathsf{same\_id\_as} \end{aligned}$$

We can still enforce the disjointness between participant properties as e.g. required to distinguish between recipient and actor. However, the disjointness of participants is not a general requirement, as agents may perform actions on themselves.

Because there is no way to automatically determine the participant roles of an action by its process description, they need to be explicitly *marked* as fulfilling a thematic role. In the most basic case, we need at least the Object_Role and Actor_Role. Their class definitions, including corresponding marker properties are as follows:

$$\begin{aligned} \mathsf{Thematic\_Role} &\sqsubseteq \mathsf{Role} \\ \mathsf{Object\_Role} &\sqsubseteq \mathsf{Thematic\_Role} \sqcap \mathsf{is\_object\_role}\ \mathbf{some\ self} \\ \mathsf{Actor\_Role} &\sqsubseteq \mathsf{Thematic\_Role} \sqcap \mathsf{is\_actor\_role}\ \mathbf{some\ self} \end{aligned}$$

Where is_object_role and is_actor_role are sub properties of the is_role marker property. Given a imposed_on assertion between an action $a$ and a process $p$, and roles attributed to the situations that are part of $p$'s causal structure, we can infer the individuals that *participate* in the action:

$$\begin{aligned} \mathsf{imposed\_on}\ \mathbf{o}\ \mathsf{is\_process}\ \mathbf{o}\ \mathsf{part}\ \mathbf{o}\ \mathsf{is\_change}\ \mathbf{o}\ \mathsf{requirement} &\sqsubseteq \mathsf{participant} \\ \mathsf{imposed\_on}\ \mathbf{o}\ \mathsf{is\_process}\ \mathbf{o}\ \mathsf{part}\ \mathbf{o}\ \mathsf{is\_change}\ \mathbf{o}\ \mathsf{result} &\sqsubseteq \mathsf{participant} \end{aligned}$$

The fillers of the actor and object properties can only be gathered by taking into account the roles they play. To accomplish this, the thematic roles need to 'backfire' to their role fillers:[24]

$$\begin{aligned} \mathsf{Object\_Role} &\sqsubseteq \mathsf{played\_by}\ \mathbf{only}\ (\mathsf{is\_object}\ \mathbf{some\ self}) \\ \mathsf{Actor\_Role} &\sqsubseteq \mathsf{played\_by}\ \mathbf{only}\ (\mathsf{is\_actor}\ \mathbf{some\ self}) \end{aligned}$$

Every individual that plays a particular thematic role is now related to itself via a marker property that indicates this role playing. The actor and object participants can now be gathered by the following property inclusion axiom:

$$\begin{aligned} \mathsf{participant}\ \mathbf{o}\ \mathsf{is\_object} &\sqsubseteq \mathsf{object} \\ \mathsf{participant}\ \mathbf{o}\ \mathsf{is\_actor} &\sqsubseteq \mathsf{actor} \end{aligned}$$

By our earlier definition of Subjective_Entity (of which Action and Role are subclasses), the subjective entity only holds within a certain context. Unfortunately, as with the utters relation in the definition of Communicated_Attitude in Section 7.3.4, we cannot specify this context via a property chain (since every subjective entity has exactly one context). However, we can again partially circumvent this problem by providing two alternate representations of a specific action_context property:

$$\mathsf{Action} \sqsubseteq \mathsf{is\_action}\ \mathbf{some\ self}$$

$$\begin{aligned} \mathsf{participant}\ \mathbf{o}\ \mathsf{plays} &\sqsubseteq \mathsf{action\_context}^- \\ \mathsf{is\_action}\ \mathbf{o}\ \mathsf{context}^-\ \mathbf{o}\ \mathsf{is\_object\_role} &\sqsubseteq \mathsf{action\_context}^- \\ \mathsf{is\_action}\ \mathbf{o}\ \mathsf{context}^-\ \mathbf{o}\ \mathsf{is\_actor\_role} &\sqsubseteq \mathsf{action\_context}^- \end{aligned}$$

[24] Of course, the relation can also be inferred in the other direction. For instance, the plays relation between some individual and its thematic role can also be inferred by specifying that e.g. every individual with an is_object relation pointing to itself plays an Object_Role.

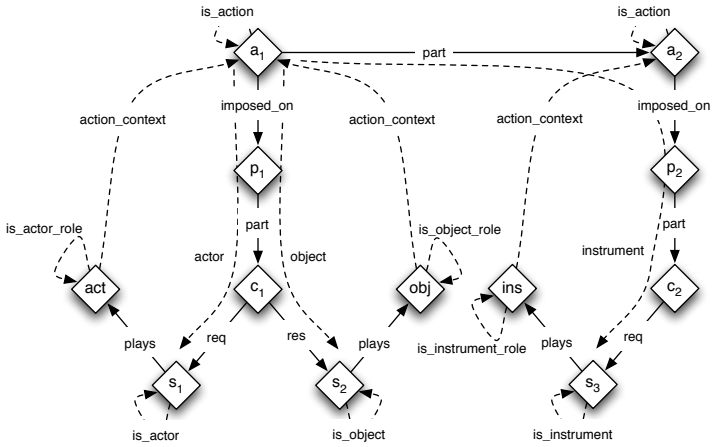


Figure 7.22: Action structure for inferring the actor, object and instrument participant roles.

The action_context property allows alternate definitions of object and actor:

$$\begin{aligned} \text{action\_context}^{-} \mathbf{o} \text{ played\_by } \mathbf{o} \text{ is\_object} &\sqsubseteq \text{object} \\ \text{action\_context}^{-} \mathbf{o} \text{ played\_by } \mathbf{o} \text{ is\_actor} &\sqsubseteq \text{actor} \end{aligned}$$

Following Schank and Abelson (1975); Breuker (1981) the instrument participant is not directly linked to the action, but is rather the object of a subordinate action. Given an explicit assertion of the plays relation between such an object and an Instrument_Role, we can infer the instrument given a composition of the part and object relations:

$$\begin{aligned} \text{Instrument\_Role} \sqsubseteq\ & \text{Thematic\_Role} \sqcap \text{is\_instrument\_role } \mathbf{some\ self} \\ & \sqcap \text{played\_by } \mathbf{only}\ (\text{is\_instrument } \mathbf{some\ self}) \end{aligned}$$

$$\begin{aligned} \text{instrument} &\sqsubseteq \text{participant} \\ \text{is\_action } \mathbf{o} \text{ part } \mathbf{o} \text{ is\_action } \mathbf{o} \text{ participant } \mathbf{o} \text{ is\_instrument} &\sqsubseteq \text{instrument} \\ \text{is\_instrument\_role } \mathbf{o} \text{ context } \mathbf{o} \text{ part}^{-} \mathbf{o} \text{ is\_action} &\sqsubseteq \text{action\_context} \\ \text{is\_action } \mathbf{o} \text{ part } \mathbf{o} \text{ is\_action } \mathbf{o} \text{ actor} &\sqsubseteq \text{actor} \\ \text{actor}^{-} \mathbf{o} \text{ part } \mathbf{o} \text{ is\_action } \mathbf{o} \text{ actor} &\sqsubseteq \text{same\_id\_as} \end{aligned}$$

The action_context of instrument roles is the action for which they are an indirect participant. The actor of the subordinate action is also the actor of the containing action, and consequently shares its identity. Figure 7.22 shows the action structure resulting from the definitions presented in this section, given a fictive action $a_1$ that consists of a process $p_1$ for which the requirement $s_1$ and the result $s_2$ of its constituting change $c_1$ are respectively the actor and object of action $a_1$. The instrument $s_2$ is the requirement of $c_2$ in process $p_2$ that forms the causal structure of action $a_2$, which is part of action $a_1$.

### An Intentional Stance

The relation between an action and its actor is somewhat harder to flesh out. We have said that actions are subjective entities imposed on a process, given the intention of some agent. At the same time, this intention is exactly to perform the action, and consequently to execute the process. However, to accommodate for the possibility that an agent fails to initiate the action he *intends* to perform, the intention and action cannot be the same thing, nor can the propositional content of the intention coincide with the process. Furthermore, where an action is objective in the sense that it does not exist merely in the mind of its actor, the intention is not.

To relate intention to action, we thus need two subjective entity summarisation triangles that both have the same actor as context. First, we represent the Intention analogous to the Belief class of Section 7.3.4:

$$\begin{aligned} \mathsf{Intention} &\sqsubseteq \mathsf{is\_intention}\ \textbf{some self} \\ &\sqsubseteq \mathsf{Propositional\_Attitude} \end{aligned}$$

$$\mathsf{holds}\ \textbf{o}\ \mathsf{is\_intention}\ \textbf{o}\ \mathsf{towards} \sqsubseteq \mathsf{intends}$$

Similarly, we summarise the relation between Agent and Process by defining an initiates role chain:[25]

$$\begin{aligned} \mathsf{performs} &\sqsubseteq \mathsf{actor}^- \\ \mathsf{performs}\ \textbf{o}\ \mathsf{imposed\_on} &\sqsubseteq \mathsf{initiates} \end{aligned}$$

In other words, the actor of an Action is said to perform it, and executes the process that counts_as the action. The Agent that executes the process is its agentive_cause, and thus counts_as the Epistemic_Role of Cause in the context of the process.

$$\begin{aligned} \mathsf{Cause} &\sqsubseteq \mathsf{is\_cause}\ \textbf{some self} \\ &\sqsubseteq \mathsf{Epistemic\_Role} \\ \mathsf{Actor\_Role} &\sqsubseteq \mathsf{played\_by}\ \textbf{only}\ (\mathsf{initiates}\ \textbf{some}\ (\mathsf{context\_of}\ \textbf{some}\ \mathsf{Cause})) \end{aligned}$$

$$\begin{aligned} \mathsf{agentive\_cause} &\sqsubseteq \mathsf{initiates}^- \\ \mathsf{executes}\ \textbf{o}\ \mathsf{context\_of} &\sqsubseteq \mathsf{imposed\_on} \end{aligned}$$

Note how the mere playing of the Actor_Role by some Agent forces it to be the initiator of some process for which it is a Cause. In the same way, the playing of this role can be used to force the Agent to hold an Intention_to_Act:

$$\begin{aligned} \mathsf{Actor\_Role} &\sqsubseteq \mathsf{played\_by}\ \textbf{only}\ (\mathsf{holds}\ \textbf{some}\ \mathsf{Intention\_to\_Act}) \\ \mathsf{Intention\_to\_Act} &\sqsubseteq \mathsf{is\_intention\_to\_act}\ \textbf{some self} \\ &\sqsubseteq \mathsf{Intention} \end{aligned}$$

This mechanism is very powerful in that it solves the inheritance problem of the counts_as relation discussed in Section 7.3.2. Subjective entities can restrict the properties of classes they are imposed on by a nested universal restriction on the counts_as property (or its sub properties).

[25] Note that this chain does not depend on the marker property for Action as the actor property already takes this into account.

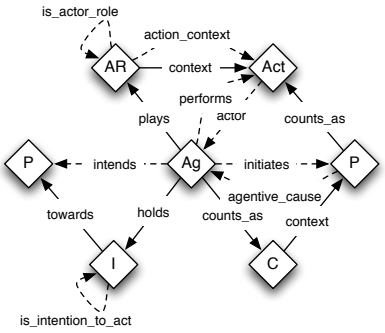


Figure 7.23: Four summarisation triangles for Action.

### 7.5.2 Discussion

The preceding section presents a representation of a complex concept, Action, by combining the three patterns described earlier. First, part of the *diamond* shape of the Transaction was extended to connect to the *sequence* pattern of Process. We had to lift the cardinality constraints on the participants in actions, to be able to define them as the *summarisation* property of the *triangular* pattern of subjective entities. This drawback is amended by adding additional same_id_as property chains, but otherwise the Transaction pattern remained in tact. The structure of the sequence pattern was used to constrain the allowed participants in an action, and identify the agentive cause of the process that counts as action.

It is rather easy, and tempting, to extend this structure even further. The composition of marker properties, roles, subjective entities, causes and actions gives an intuitive handle on the definition of more complex social constructs in LKIF Core, such as legal notions related to liability. Following Lehmann (2003) the agent that initiates a process which causes an event that results in some damage can be said to be *potentially liable* for that damage. Without giving a full definition of these concepts, a straightforward definition of this relation is expressible as a role chain that uses the caused_by, agentive_cause and played_by properties:

$$\mathsf{Damage} \sqsubseteq \mathsf{Legal\_Role} \sqcap \mathsf{played\_by}\ \mathbf{only}\ (\mathsf{is\_damage}\ \mathbf{some}\ \mathbf{self})$$

$$\mathsf{is\_damage} \circ \mathsf{result\_of} \circ \mathsf{caused\_by} \circ \mathsf{agentive\_cause} \sqsubseteq \mathsf{potentially\_liable}$$

The four summarisation triangles involved in the intentional perspective on actions are depicted in Figure 7.23.

## 7.6 Discussion

This chapter presented and discussed three design patterns inspired by, but not limited to the LKIF Core ontology presented in the previous chapter. Rather than *content* patterns, these emphasise the *structure* common to many representations. The first pattern, of Section 7.2, is inspired by the general inability

to constrain the models of a DL class description in such a way that they correspond to a *diamond*-like structure (Section 7.1.1). A typical use case for this pattern is the notion of *exchange*, but similar application can be found in the description of physical, structured objects, such as the human heart, and artefacts, such as tables and electronic devices. Whether the resulting class description is an *ontological* definition or rather a *framework* depends on its object, rather than its structure.

The design pattern was introduced in an incremental fashion, emphasising the procedural nature of design, and elucidating the decisions and trade-offs involved in the representation of the Transaction class. The procedure is composed of five steps: creating an initial class description, constrain the number of role fillers, disambiguate between role fillers, traverse the tree model of the class definition using property chains, and introduce asymmetry between the branches of the tree to allow for precise property chains. Central to this procedure is to ensure first the recognition of individuals belonging to a class, and secondly enforce compliance to the class definition when an individual is asserted to be a member. Once the definition is as strict as possible, additional information is *inferred* by the liberal use of subclass axioms and, in particular, role inclusion axioms.

The pattern introduces the same_id_as and different_id_from properties as alternatives to the built-in owl:sameAs and owl:differentFrom properties. Because the latter are part of the OWL semantics, they are subject to the tree model restriction of DL. However, the former are not and can be interpreted in two custom ways: as equivalent to their built-in counterparts, e.g. by specification of the DL-safe rule in Section 7.2.2, or as carrying the *identity* of the individuals it relates. The third pattern, of Section 7.4 illustrates that this second interpretation is particularly useful in the description of different *states* of a single entity. The use of nominals or value restrictions as placeholders in class descriptions is not a general alternative to the pattern described here as its applicability is restricted to controlled environments (see Section 7.2.2).

Section 7.3 presents a *triangular* pattern for the summarisation of reified relations, based on use cases for n-ary relations identified by the SWBP (Noy and Rector, 2006), and common ways to define social concepts. The same cannot be achieved using standard RDF reification. The main contributions of this pattern are the ability to construct and combine complex relations using simple combinations of classes and roles, and a general method for abstracting from expressive ontologies to create a simplified version that is easier to reuse.

The pattern is constructed largely by the same procedure as that of Section 7.2, but extends it with a step to introduce *domain dependence* of the summary property, and an optional *punning* step. It is characterised by the use of *marker properties* to identify classes from within the RBox, and *backfiring* of property restrictions over the relations. The latter mechanism is of particular interest to social and legal domains where role playing and e.g. deeming provisions rely heavily on the inheritance of properties from anti-rigid classes over the counts_as relation, as opposed to standard subsumption. In addition to the same_id_as mechanism of the first pattern, this pattern furthermore demonstrates a mechanism for *wrapping* a simple property inside a role inclusion to simulate a rdfs:subPropertyOf relation. This to circumvent situations where asserting this relation would make the property complex (see Section 7.3.4).

The third design pattern, described in Section 7.4, is an approach to repres-

enting *sequences* inspired both by requirements for causal explanation in law (Hoekstra and Breuker, 2007), and protein sequences in biomedical research (Drummond et al., 2006). The construction of class definitions using this pattern follows a similar methodology as that of Section 7.2 and Section 7.3. However, it differs in that membership of the sequence class depends entirely on the inferred property chain, where the other patterns primarily use property chains to infer new relations. It is very lean and flexible compared to existing approaches as sequences are defined solely by role inclusion axioms. An important benefit is that role inclusions can easily be combined to form more elaborate chains.

The combination of the three patterns in Section 7.5 shows that not only the patterns, but perhaps more importantly the *techniques* central to the patterns can be usefully applied to construct elaborate class definitions. It turns out that extension of the three design patterns can be conservative without compromising significant expressive power. This is because they are self-contained and their synthesis is only defined by means of property inclusion axioms and additional class *definitions*, rather than *restrictions* on existing classes. The principle of *backfiring* is a good example; using the subjective entity summarisation pattern, we can specify conditional restrictions on the instances of a particular class without affecting the original class definition. For instance, the Actor_Role requires every Agent that plays the role to initiate some Process, but does not alter the definition of the Agent class itself. In fact, it does not even mention it. Loose coupling of the three patterns furthermore corroborates the assumption of stratification that underlies the distinction between the different levels of description in LKIF Core (see Section 6.3).

As a general approach to ontology development, the procedures used in the patterns are limited in several ways. We have seen multiple times that property inclusions are sub properties of the property they define (Section 7.4.3). This means that although they may allow us to infer the existence of a property in more cases than before, it does not affect the pre-existing semantics of that property provided that it is already treated as *complex*. The use of such extensible properties must therefore follow the global restrictions defined in Motik et al. (2009). It is advisable that ontology engineers who employ one of the patterns explicitly mark simple properties intended for extensibility using some annotation property, similar to e.g. the meta-properties proposed by Guarino and Welty (2004) for marking the rigidity of a class (see Section 5.5.1).

A related restriction is that the assertion of individuals should take the delicate semantics of role inclusion axioms into account: asserting a value for a property defined by such a chain does not imply the existence of that chain between an individual and the value. In principle, this limitation is amendable by introducing macro-like named property chains that allow assertions and class restrictions to refer to the chain directly, rather than its super property. However, such named chains would be subject to strict global restrictions to prevent undecidability.

The third restriction follows from the custom solution to the limited expressiveness of OWL 2 in describing non-tree like structures. The same_id_as and different_id_from properties carry special semantics that is not part of OWL 2 itself, but can be very useful in the context of a knowledge based system that has to deal with the manipulation of multiple entities in different states. An ontology engineer will have to make a conscious decision to interpret them either

as equivalent to their built-in counterparts, or as identity-carrying properties.

On the other hand, the design patterns presented in this chapter demonstrate a promising avenue for a more liberal reuse of parts of ontologies. Rather than capturing a strict ontological commitment, design patterns express an ontological perspective, that can be applied to basic concepts to construct derivative notions.[26]

[26]The patterns described in this chapter are available as OWL files at `http://ontology.leibnizcenter.org`.

# Chapter 8

# Conclusion

> "Whoever wishes to foresee the future must consult the past; for human events ever resemble those of preceding times. This arises from the fact that they are produced by men who ever have been, and ever shall be, animated by the same passions, and thus they necessarily have the same results."
>
> *Machiavelli*

## 8.1 Introduction

This book presents an exploration of lessons learned in over forty years of research in the field of knowledge based systems. Chapter 2 covers the period from the sixties until the early nineties. During these years, increased experience in using knowledge representation to build intelligent applications culminated in the awareness of two important requirements: the need to separate different types of knowledge, and to develop formal, restricted knowledge representation languages tailored for each of these types. Jointly these constitute a *knowledge representation* perspective that has far reaching consequences for how knowledge representation artefacts are built.

Seven consecutive chapters explore these consequences for knowledge that reflects the ontological commitments of a system's domain theory. The specification of such *ontologies* is an important step in the *knowledge level* specification of knowledge based systems (see Section 2.4). A guarantee on the computational properties of terminological representation languages allows ontologies to be used directly as knowledge system components (Section 2.5). Ontologies that are meant to be used in this way are *knowledge representation* ontologies (Section 4.5), and are typically represented using languages belonging to the family of description logics. Chapter 3 discusses how a highly expressive description logic has been crafted on top of existing languages for the web, and

introduces the resulting language: OWL 2 (Motik et al., 2009). Chapter 4 investigates and disentangles different uses of the word 'ontology' and presents a clear view of the role of ontologies in knowledge based systems.

Chapter 5 discusses a selection of methodologies and principles for ontology development that have seen the light over the past fifteen years. Chapters 4 and 5 jointly provide a framework of requirements for ontology development in the context of knowledge representation. These requirements are further refined in Chapter 6, which describes the LKIF Core ontology of basic legal concepts. In particular, the consequences of the knowledge representation perspective are discussed in the relation between LKIF Core and existing ontologies. Finally, Chapter 7 brings together chapters 3 and 6 in a confrontation with the OWL 2 DL knowledge representation language. The design patterns presented there are representations of concepts and structures that can be found across many domains, and follows from an exploration of the full extent of OWL 2 DL's expressiveness. Their description conveys a skeleton methodology for constructing OWL class definitions, that makes use of several novel micro patterns.

In the introduction of this book I formulated five questions related to the task of knowledge representation:

- *Quality* – How can the quality of models be improved?
- *Design* – Can the design of models be facilitated, or made easier?
- *Compatibility* – To what extent do theory and practice go hand in hand?
- *Rationale* – What is the rationale behind representation languages?
- *Expressiveness* – How do limitations in expressiveness affect models of a concrete domain?

The following sections discuss the findings from chapters 2 through 7 in relation to these questions, and present a number of conclusions.

## 8.2 Compatibility of Ontologies

Ontology development is an important part of a knowledge systems development methodology that relies on a maximisation of *knowledge reuse*. Indeed, existing ontologies are often used as bootstrap for the creation of larger ontologies, or as axiomatic foundation for more extensive knowledge representations. This ontology reuse is hindered by two important factors. Firstly, ontologies can be overwhelming in complexity and size; they are hard to interpret and extend. Secondly, direct ontology reuse is subject to the *ontology interaction problem* identified in Section 5.4.2: the problem that axioms in a reused ontology and its extension may interact in unpredictable and undesirable ways, causing the semantics of a reused ontology to change. For expressive languages, no decidable algorithm exists that can determine whether the extensions of an ontology are mutually compatible with respect to its axiom closure.

### Soft Reuse

The most common solution to this problem is to limit the role of axioms. Either by shifting attention to *lightweight* ontologies, or by maintaining a *less restrictive* definition of reuse. The first approach is apparent in contemporary work on knowledge acquisition (Gangemi and Euzenat, 2008). For instance, complex tasks such as ontology merging and alignment are considered reducible to linguistic matching of concept names and distance measures over (taxonomic) relations. However, this is undesirable from a knowledge representation perspective: lightweight ontologies cannot account for the domain theory in knowledge based systems. The second approach is exemplified by allowing 'repairs' to reused ontologies, e.g. by proposing ways to alleviate semantic mismatches (discussed in Section 5.4). It is hard to see how this approach contributes to the compatibility and reusability of systems, components and services.

Both approaches sacrifice the technical reusability of knowledge components for a reusability claim at a more conceptual level: two systems that 'reuse' the same ontology may be incompatible in its 'use'. Although indeed this suggests that the original ideal of fully compatible knowledge systems may have been somewhat naive, it remains that the number of incompatibilities should be minimised, by maximising reusability of ontologies intended for that purpose.

### Perspectives

This last observation inspired the idea to design ontologies for specific types of reuse, and has become a central part of most ontology engineering methodologies (Section 5.4.1). For instance, the distinction of van Heijst et al. (1997) between top, core, domain and application ontologies was meant to organise ontology libraries in a modular fashion, with strictly separate levels of description. However, these distinctions are rarely applied in practice, as immediate needs often prevail over methodological principles: existing ontologies tend to freely mix levels of description. Furthermore, the drawback of such categorisations is that they do not capture the purposes for which an ontology has been developed. As I discuss in Chapter 4, existing ontologies tend to mix three very distinct perspectives (Section 4.5):

- *Knowledge Management Ontologies* are (structured) vocabularies developed for sharing and reuse of information within organisations.
- *Knowledge Representation Ontologies* are reusable terminological knowledge representations that specify the part of a domain theory of knowledge based systems that directly reflects its ontological commitment.
- *Formal Ontologies* are formal specifications of an ontological theory in philosophy.

In chapter 4 and 5, I argue that it is the compatibility between these three perspectives that determines whether two ontologies are *in principle* compatible. A commitment to either of these perspectives entails the adoption of a set of specific requirements (Section 5.4.3).

### Language

In Section 5.4.2, I discuss a formal framework for assessing the type of reuse possible between two ontologies. Section 5.4.3 formulates five restrictions that apply to ontology reuse in the context of knowledge sharing between knowledge based systems. In particular, reuse is governed by *language compatibility*, which has a direct effect on the reusability of ontologies across the three perspectives. The level of language expressiveness required to accommodate each view ranges from featherweight to first order logic. For instance, the specification of a knowledge management ontology does not require an expressive language, as it will be used only for limited forms of reasoning. For formal and knowledge representation ontologies, on the other hand, more expressiveness is desirable as it increases their adequacy as models of reality. Language compatibility is glossed over by existing methodologies: they invariably adopt a weaker interpretation of reuse.

At first sight, the latter two ontology types seem to have corresponding requirements. However, as outlined in Chapter 2, if an ontology is to be used as knowledge component in a reasoning architecture, it should be restricted to formalisms that take into account the *restricted language thesis* of Levesque and Brachman (1987): inference over the language should be efficient and decidable (see Section 2.5). In other words, it should be dependable and produce correct answers within a predictable amount of time. As a result, there exists a trade-off between expressiveness, i.e. potential ontological adequacy, and utility: the scope of knowledge representation ontologies is determined by a hard limit.

For formal ontologies in philosophy, as with other predominantly theoretical disciplines, such limitation given by practical considerations is not pertinent to their purpose. Nonetheless, even an ontology as formal consolidation of a metaphysical theory is subject to the structural limitations of formal language (see Section 4.3.1). Formalisation is not a silver bullet: no matter how expressive a formal language is, a formal theory will never fully cover reality. The current state of the art in the field of knowledge representation results from the acknowledgement of this fact, and ensures that at least we can apply our approximate theories to something *useful*, other than only the advancement of scientific thought. As long as this fundamental trade-off is not acknowledged by philosophical ontologists, a usefulness claim of formal philosophical ontologies for knowledge based systems remains rather dubious.

### Scope and Abstraction

The LKIF Core ontology, described in Chapter 6, is a knowledge representation ontology. It is developed as part of an effort to provide a common vocabulary for information exchange between legal knowledge based systems. Section 6.2 discusses four formally specified, generic ontologies and considers them for reuse and integration in LKIF Core: SUMO (Niles and Pease, 2001), DOLCE (Masolo et al., 2003; Gangemi et al., 2002), CLO (Gangemi et al., 2005), and CYC (Lenat et al., 1990; Lenat, 1995). The discussion shows that these ontologies are not reusable for two reasons. First, their level of generality is obtained in a large part by the use of higher order abstractions that cannot be expressed in even the most expressive decidable knowledge representation languages: they do not meet the language compatibility requirement. Secondly, the perspect-

ive of an ontology determines its *scope* and level of *abstraction* (Section 5.4.1). This makes that the level of description of a generic formal ontology may well be applicable to a certain domain, but is of only limited help in the definition of more concrete concepts used in reasoning: they do not facilitate knowledge acquisition. A similar limitation resides in methodologies that emphasise abstract ontological principles, such as ONTOCLEAN (Section 5.5.1). Although practical for keeping track of, and justifying design decisions, such principles do not directly help solve concrete knowledge acquisition problems.

#### Common Sense

The LKIF Core ontology takes a different approach: rather than a focus on the abstractions that *transcend* multiple domains (as in philosophy), the ontology represents notions they have in *common*. It is a *core ontology*, and aims to give definitions of concepts that are *basic* to both law and common sense. The focus on commonalities, and cognitively basic notions, is a key part of ontology engineering methodologies developed during the nineties (Section 5.3).

LKIF Core takes this one step further and emphasises the origins of common sense as an important inspiration for the structure of the ontology. This consideration of human cognition is reminiscent of the early days of knowledge representation, and in particular the use of semantic networks to represent semantic memory (Section 2.2.3). However, the current focus is far less ambitious: the correspondence between representation and cognition is taken as facilitator for acquisition, usability and reuse, rather than as corroboration of a cognitive psychological theory of human intelligence. In the context of the Semantic Web, the ontology aligns with the perspective of a human user, rather than with just the technical perspective of machine-machine communication.

#### Frameworks

The quest for commonalities easily derails in the definition of recurrent structures that are generic across situations, but are not of an ontological nature. While subsumption is central to ontology specification, its use is not reserved to ontologies. For instance, in typologies that capture permutations of property values rather than proper ontological categories. Section 5.5.2 discusses the distinction between ontologies and three types of *frameworks*: situational, mereological, and epistemological frameworks. These frameworks are similar to Schankian scripts in that they capture the *context* of ontological categories; i.e. their accidental rather than intrinsic properties. They rely on partonomy and dependency relations; two of the three relations considered primary by Smith (2004); Breuker and Wielinga (1987).

## 8.3 Quality and Design

Over the years, significant effort has been directed to facilitate the ontology design process. As ontologies became increasingly specified using formal languages, the *engineering* view on knowledge based systems development and design, which emerged in the early nineties, was applied the development of

ontologies as well: methodologies, design principles, tools and – more recently – design patterns each aim to provide handles for ontology construction.

### Design Patterns

We have seen that ontology engineering methodologies position the representation of an ontology as part of a relatively opaque 'ontology coding' step. In part this lack of detail is remedied by the incorporation of available existing ontologies, but they can only partly contribute to the quality of a new ontology, given the limitations on ontology reuse discussed in the previous section. Interest therefore grows in extracting the parts of ontologies that can be usefully applied in the construction of new ontologies. The recent rise in popularity of such *design patterns* marks the start of a new development in ontology construction, aimed at sharing best practices, rather than ready-made solutions. A second reason for investigating design patterns is the expectation that they would allow for a more lightweight form of ontology reuse.

However, the discussion of design patterns in Section 5.6 shows that the ontology interaction problem holds for most of these patterns as well, and *content* ontology design patterns in particular, as they are wholly defined in terms of domain concepts. On the other hand, logical design patterns cannot play the same role as they are specified at a meta level comparable to that of the language constructs themselves. In Section 5.6.4, I therefore propose the *metaphoric* use of *structure* patterns that are less restrictive with respect to their implementation in ontologies.

### Micro Patterns

A closer inspection of the internal structure of design patterns, reveals that they need not necessarily be directly specified at the level of OWL constructs: a number of intermediate, composite structures appear across the three design patterns described in Chapter 7. The techniques for role-chain summarisation, identity and difference propagation, backfiring of property values, the combination of self-restrictions and role chains, and wrapping properties in chains are concrete tools for ontology representation in OWL. Such *micro-patterns* reflect modelling techniques, rather than concrete patterns that need to be transposed when applied to a new domain (cf. Section 5.6.4). Micro-patterns are good candidates for extensions to OWL editors, either as macros (proposed by Vrandečić (2005)), or as simple plugins. For instance, the AddMarkers plugin for Protégé 4 automatically adds marker properties to all classes in the active ontology.[1] It is not unlikely that some of these micro patterns are common enough to warrant future addition to the OWL syntax as 'syntactic sugar'.

### Patterns in Design

Design patterns are the result of a *design task*, a description of this design process conveys their rationale and improves accessibility. The sharing of design patterns should take into account how and why they are created: rather than patterns in ontology, design patterns should capture patterns in *design*. One

[1] See `http://www.leibnizcenter.org/users/rinke/2008/10/08/addmarkers-protege-4-plugin/`

step beyond structure and micro patterns is therefore the conscientious collection and dissemination of *experience* in creating ontology content.

The previous chapter introduces three patterns, which are deliberately described in terms of the steps needed to apply them to concrete domain use cases. In fact, the overlap in these steps indicates that ontology construction involves the application of certain representation strategies that in conjunction fulfil a design task. Much akin to the way in which problem types index the library of problem solving methods of Breuker (1994), this suggests that use cases for design patterns should be cast in terms of design tasks, rather than exclusively in domain terms. Chapter 7 can be regarded as a preliminary study of this approach and I believe a more rigorous investigation in this direction will benefit the quality and design of ontologies to a much larger extent than even the most elaborate library of content ontology design patterns will ever do.

## 8.4 Rationale and Expressiveness of OWL 2

The restricted nature of knowledge representation languages such as OWL 2 DL is given by technical, computational considerations of chapters 2 and 3, rather than the theoretical and methodological issues outlined in chapters 4 and 5. We have seen ample evidence that this language is not a direct 'fit' for ontology representation: not everything that is an ontology can be expressed in DL, and not everything that is expressible is an ontology. While the latter problem can be responded to using a strict methodology and adherence to principles, the former is more problematic. The knowledge representation perspective puts reuse before use: why should I use a language that does not allow me to express what I need for my application?

#### Decidability

Although many of the arguments of Doyle and Patil (1991) against the restricted language thesis of Levesque and Brachman (1987) have been superseded – simply because description logics have become much more expressive – decidability is still very topical. For instance, the distinction between OWL DL and OWL Full was introduced because of disagreement within the WebONT working group as to whether decidability was a must-have feature for OWL. The call for expressive languages is often legitimate, e.g. given theoretical considerations (in philosophy), as well as where applications require undecidable (or unexplored) extensions such as functional datatype properties ('keys') or variables. However, a resolve not to use OWL 2 DL because of its limitations is not always justified, and the choice for a more expressive language may harbor hidden surprises. Although decidability may be a rather harsh requirement in itself, it does offer dependability.

There is a hard limit to what can be expressed in OWL 2 DL, and to a large extent this limitation is the result of a commitment to decidable reasoning. However, this is not the only reason. The historic review in Chapter 2 shows that separating different types of knowledge is crucial to the success of maintainable and reusable knowledge components. This need was the main motivation for the development of languages targeted to the representation of distinct knowledge types. Description logics were inspired by the need to provide a

formal basis for the frame-paradigm, used to capture the terminological knowledge used in a system's domain theory (Section 2.3.2). As a consequence, DL reasoners are optimised for performing inference tasks typical to this type of knowledge: classification (TBox) and realisation (ABox).

Decidable rule languages may have other, enticing expressiveness bounds, but this comes at the cost of constructs that OWL 2 DL *does* cater for, such as e.g. cardinality restrictions (cf. the OWL 2 RL profile in Section 3.5.2). Furthermore, it is to be expected that the interaction problem between knowledge resources is even more significant for rule languages. Firstly, as these are not tailored for a specific type of knowledge, the knowledge *types* in a rule base more easily interact (Bylander and Chandrasekaran, 1987). Where description logics have come forth from a need to perfect the terminological aspects of domain knowledge, rule languages have never been optimised for e.g. representing strategic or causal knowledge in the same way.[2] Secondly, the semantics of rules is more likely to interact than that of OWL 2 DL axioms, as the latter has been specifically designed to allow for extensible representations: rule languages adopt the closed world assumption, where OWL assumes an open world (Section 3.4).

#### Perception

A principal conclusion of Chapter 7 is that OWL 2 DL is expressive enough for many practical problems. The specification of transactions, roles, processes and actions in OWL 2 DL shows that by using the language in novel ways, the definition of concepts can be sufficiently precise, with acceptable trade-offs. In fact, I argue that in many cases the inexpressiveness of DL is *perceived*. For instance, the pattern for n-ary relations in Section 7.3 shows that this impression is often based on *syntactic* considerations.

The mismatch between perceived and actual expressiveness of OWL is a result both of its innate complexity, and of the fact that it differs significantly from other expressive, intuitive, but sometimes under specified languages. This brings us to one of the main the arguments for favouring a rule language over a description logic: people are often considered to 'think in rules'. Although the call for an intuitive language is certainly justifiable, this is an interface issue. In fact, there currently exist several alternative presentations for DL axioms: structured natural-language syntaxes for OWL (Kaljurand and Fuchs, 2007; Cregan et al., 2007; Schwitter et al., 2008), the visual editing of OWL ontologies proposed by Brockmans et al. (2006), and even proposals to present DL axioms as rules (Gasse et al., 2008; Krötzsch et al., 2008a). Clearly, an intuitive presentation syntax does not necessarily correspond to appropriateness for automated reasoning.

## 8.5 Closing Remarks

I hope this book contributes to a better understanding of why ontologies are useful, and why a language such as OWL 2 DL is its quirky little self. For a long time, ontologies were seen as the geese that lay the golden eggs. Now that the shine is slowly wearing off, the danger is that they are dismissed as

[2] In fact, the representation of causal knowledge is the main focus of languages in qualitative and model-based reasoning, see Section 7.4.1.

slogan level management-speak, or exiled to the realm of metaphysical debates between philosophers. In this book, I have tried to show that ontologies are no different than other knowledge representation artefacts. Their applicability may be restricted only to certain cases, but they still play a vital role in any serious endeavour to the development of reusable knowledge based services, especially when these are published on the web.

A related observation is that web-based – and certainly web-scale – reasoning is still an open issue. Even relatively lightweight applications, such as distributed querying of RDF resources and the composition of semantic web services have a long way to go. In some sense this is disappointing, as the longer the promise of a true *semantic* web is pending, the more likely it is that raw power data mining approaches claim some of its potential market share. Nonetheless, the very idea of a semantic web already sparked an enormous variety of novel technologies that can be used for other, more modest applications. OWL 2 DL is not just a web language; it is a powerful and expressive language for building knowledge based components as well. The ability to share ontologies across the web is a huge improvement, and solves many of the issues that concerned the knowledge acquisition community during the early nineties (i.e. ontology libraries). OWL could not have reached its current status and user base without its foundation in the ideal of a Semantic Web.

However, we must be careful not to default too easily to OWL 2 DL as ontology representation language. For, is OWL 2 useful for representing ontologies in general? The extensive discussion on the various perspectives on ontologies leads me to conclude that this is certainly not always the case. If you are committed to building an ontology that is to be used for (decidable) knowledge based reasoning, and can be shared on the web, it is certainly the most advanced language currently available. For building a philosophical foundational ontology, a language with minimal ontological and epistemological bias such as predicate logic may be more suitable. However, this will render the ontology unsuitable for automated reasoning in practical systems. Thirdly, if the main purpose is the standardisation and sharing of a *vocabulary*, a less expressive language is more suitable (Section 3.3). It is good to see that the need for well designed, but lightweight languages is increasingly being catered for; both through the introduction of language profiles in OWL 2, and in the SKOS vocabulary definition language.

It is my firm belief that the status granted to ontologies during the nineties was overzealous, and although it paved the way for the adoption of a well wrought language as standard for knowledge representation on the web, it also instigated a lot of confusion that led both to entrenched positions in scientific debate and to disappointment of early adopters. In this book I question the claim that ontologies based on philosophical theories are somehow 'better', which is implicit in many publications on formal Ontology (e.g. Guarino and Giaretta (1995); Bera and Wand (2004); Barry Smith (2007)). I argue that this claim be evaluated in the context of an ontology's purpose, and that for knowledge representation ontologies more practical requirements take precedence. This insight forms the backbone of the LKIF Core ontology and design patterns presented here, and I believe it is a viable alternative to the philosophical approach. Nonetheless, an empirical study as to which approach is more appropriate under various circumstances would certainly be more conclusive.

We have seen that committing to a knowledge representation perspective

exposes ontology development to inherent trade-offs and problems, and requires a thorough understanding of the knowledge representation language used. Ontology development consists of two phases, first knowledge is extracted from experts, then this knowledge is expressed using a formal language. The caveat is the medium through which this takes place: the ontology engineer. Rather than expert knowledge, it is *his* knowledge that is captured in formal representations: the engineer is the true knowledge acquisition bottleneck. This realisation is lacking in both ontology engineering methodologies and traditional design patterns, at least to the extent that they cover only part of the ontology development process.

This book covers both theory and practice of knowledge acquisition, representation and ontologies; it emphasises human understanding as knowledge structuring principle, and demonstrates this approach in the development of an ontology, and the description, justification and representation of several design patterns. In doing so I hope it contributes to a better understanding of the representation of ontologies; or rather, what it means to *do* ontology representation.

# Bibliography


Agnoloni, T., Bacci, L., Francesconi, E., Peters, W., Montamegni, S., and Venturi, G. (2009). A two-level knowledge approach to support multilingual legislative drafting. In Breuker, J., Casanovas, P., Klein, M., and Francesconi, E., editors, *Law, Ontologies and the Semantic Web*, Frontiers of Artificial Intelligence and Applications. IOS Press, Amsterdam.

Allen, J. (1984). Towards a general theory of action and time. *Artificial Intelligence*, 23:123–154.

Allen, J. F. and Ferguson, G. (1994). Actions and events in interval temporal logic. Technical Report TR521, Rochester University, Department of Computer Science, NY.

Anderson, J. and Bower, G. (1973). *Human associative memory*. Winston & Sons, Washington, DC.

Antoniou, G. and van Harmelen, F. (2003). Web ontology language: OWL. In *Handbook on Ontologies in Information Systems*, International Handbooks on Information Systems, pages 67–92. Springer.

Antoniou, G. and van Harmelen, F. (2004). *A Semantic Web Primer*. MIT Press.

Baader, F., Brandt, S., and Lutz, C. (2005). Pushing the $\mathcal{EL}$ envelope. In *Proceedings on the 19th Joint International Conference on Artificial Intelligence (IJCAI 2005)*.

Baader, F., Bürckert, H.-J., Heinsohn, J., Hollunder, B., Müller, J., Nebel, B., Nutt, W., and Profitlich, H.-J. (1991). Terminological knowledge representation: A proposal for a terminological logic. In *Description Logics*, pages 120–128.

Baader, F., Calvanese, D., McGuinness, D., Nardi, D., and Patel-Schneider, P. F., editors (2003). *The Description Logic Handbook - Theory, Implementation and Applications*. Cambridge University Press.

Baader, F. and Hanschke, P. (1991). A scheme for integrating concrete domains into concept languages. Technical Report RR-91-10, Deutsches Forschungszentrum für Künstliche Intelligenz GmbH, Kaiserslautern, Germany.

Baader, F. and Hollunder, B. (1991). Kris: Knowledge representation and inference system. *SIGART Bull.*, 2(3):8–14.

Barry Smith, e. a. (2007). The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration. *Nature Biotechnology*, 25(25):1251–1255.

Bechhofer, S., Harper, S., and Lunn, D. (2006). SADIe: Semantic annotation for accessibility. In *5th International Semantic Web Conference*, Athens, GA, USA. LNCS 4273.

Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D. L., Patel-Schneider, P. F., and Stein, L. A. (2004). OWL web ontology language reference. W3C recommendation, World Wide Web Consortium (W3C). M. Dean, G. Schreiber (eds.).

Bench-Capon, T. and Coenen, F. (1991). Isomorphism and legal knowledge based systems. *Artificial Inteligence and Law*, 1(1):65–86.

Bera, P. and Wand, Y. (2004). Analyzing OWL using a philosophy-based ontology. In Varzi, A. C. and Vieu, L., editors, *Formal Ontology in Information Systems, Proceedings of the Third International Conference (FOIS 2004)*, volume 114 of *Frontiers in Artificial Intelligence and Applications*. IOS Press.

Berners-Lee, T. (1999). *Weaving the Web: Origins and Future of the World Wide Web*. Texere Publishing.

Berners-Lee, T., Hendler, J., and Lassila, O. (2001). The Semantic Web. *Scientific American*, 284(5):34–43.

Biezunski, M., Bryan, M., and Newcomb, S. R. (1999). Topic maps: Information technology – document description and markup languages. Technical Report 13250:2000, ISO/IEC.

Bobillo, F., Delgado, M., and Gómez-Romero, J. (2007). An ontology design pattern for representing relevance in OWL. In *Proceedings of the 6th International Semantic Web Conference (ISWC 2007)*, Busan, Corea.

Bobrow, D. G. and Winograd, T. A. (1976). An overview of KRL, a knowledge representation language. Technical report, Xerox Palo Alto Research Center, Stanford, CA, USA.

Bodenreider, O., Smith, B., and Burgun, A. (2004). The ontology-epistemology divide: A case study in medical terminology. In Varzi, A. C. and Vieu, L., editors, *Formal Ontology in Information Systems, Proceedings of the Third International Conference (FOIS 2004)*, Frontiers in Artificial Intelligence and Applications, pages 185–195, Torino. IOS Press.

Boer, A. (2000). The Consultancy Game. In Breuker, J., Leenes, R., and Winkels, R., editors, *Legal Knowledge and Information Systems. Jurix 2000: The Thirteenth Annual Conference*, Frontiers in Artificial Intelligence and Applications, pages 99–112, Amsterdam. IOS Press.

Boer, A. (2006). Note on production rules and the legal knowledge interchange format. Technical report, Leibniz Center for Law, Faculty of Law, University of Amsterdam.

Boer, A. (2009). *Legal Theory, Sources of Law & the Semantic Web*. PhD thesis, Faculty of Law, Universiteit van Amsterdam. To be published.

Boer, A., Gordon, T. F., van den Berg, K., Di Bello, M., Förhécz, A., and Vas, R. (2007a). Specification of the legal knowledge interchange format. Deliverable 1.1, Estrella.

Boer, A., Hoekstra, R., Winkels, R., van Engers, T., and Willaert, F. (2002). $^{META}lex$: Legislation in XML. In Bench-Capon, T., Daskalopulu, A., and Winkels, R., editors, *Legal Knowledge and Information Systems (Jurix 2002)*, pages 1–10, Amsterdam. IOS Press.

Boer, A., van Engers, T., Peters, R., and Winkels, R. (2007b). Separating law from geography in GIS-based egovernment services. *Artificial Intelligence & Law*, 15(1):49–76.

Boer, A., van Engers, T., and Winkels, R. (2005a). Mixing legal and non-legal norms. In Moens, M.-F. and Spyns, P., editors, *Jurix 2005: The Eighteenth Annual Conference.*, Legal Knowledge and Information Systems, pages 25–36, Amsterdam. IOS Press.

Boer, A., van Engers, T., and Winkels, R. (2005b). Open standards for spatial regulations: an interdisciplinary approach. In *Proceedings of the Holland Open Software Conference*, Amsterdam, Netherlands.

Boer, A., Winkels, R., and Hoekstra, R. (2001). The CLIME Ontology. In Winkels, R. and Hoekstra, R., editors, *Proceedings of the Second International Workshop on Legal Ontologies (LEGONT)*, pages 37–47, Amsterdam, Netherlands.

Boer, A., Winkels, R., van Engers, T., and de Maat, E. (2004a). A content management system based on an event-based model of version management information in legislation. In Gordon, T., editor, *Legal Knowledge and Information Systems. Jurix 2004: The Seventeenth Annual Conference.*, Frontiers in Artificial Intelligence and Applications, pages 19–28, Amsterdam. IOS Press.

Boer, A., Winkels, R., van Engers, T., and de Maat, E. (2004b). Time and versions in $^{META}lex$ XML. In *Proceeding of the Workshop on Legislative XML*, Kobaek Strand.

Boer, A., Winkels, R., and Vitali, F. (2007c). Proposed XML standards for law: Metalex and LKIF. In Lodder, A. R. and Mommers, L., editors, *Legal Knowledge and Information Systems. Jurix 2007: The Twentieth Annual Conference Annual Conference*, volume 165 of *Frontiers in Artificial Intelligence and Applications*, pages 19–28. IOS Press.

Borgida, A., Brachman, R., McGuinness, D., and Resnick, L. (1989). Classic: A structural data model for objects. In *Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data*, Portland, OR. ACM Press.

Brachman, R. (1979). On the epistemological status of semantic networks. In Findler, N., editor, *Associative Networks*, pages 30–50. Academic Press, New York.

Brachman, R. (1983). What IS-A is and isn't. *IEEE Computer*, 16(10):30–36.

Brachman, R. and Schmolze, J. (1985). An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2):171–216.

Brachman, R. J., McGuiness, D. L., Patel-Schneider, P. F., Resnick, L. A., and Borgida, A. (1991). Living with CLASSIC: When and how to use a KL-ONE-like language. In Sowa, J. F., editor, *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, pages 401–456. Morgan Kaufmann, San Mateo, US.

Bratko, I. (1986). *Prolog Programming for Artificial Intelligence*. Addison-Wesley, Reading, MA.

Bredeweg, B. (1994). Model-based diagnosis and prediction of behaviour. In Breuker, J. and de Velde, W. V., editors, *CommonKADS Library for Expertise Modelling*, pages 121–153. IOS-Press/Ohmsha, Amsterdam/Tokyo.

Bredeweg, B., Salles, P., and Neumann, M. (2006). *Ecological Informatics: Scope, Techniques and Applications*, chapter Ecological Applications of Qualitative Reasoning, pages 15–47. Springer, Berlin, 2nd edition.

Bredeweg, B. and Struss, P. (2004). Current topics in qualitative reasoning. *AI Magazine*, 24:13–16. special issue on Qualitative Reasoning.

Breuker, J. (1990). Towards a Workbench for the Legal Practitioner. In van Noortwijk, C., Schmidt, A., and Winkels, R., editors, *Legal Knowledge Based Systems: Aims for Research and Development, JURIX–1990*, pages 25 – 36, Lelystad. Koninklijke Vermande BV.

Breuker, J. (1994). *CommonKADS Library of Expertise Modelling*, chapter A Suite of Problem Types, pages 57 – 88. IOS-Press/Ohmsha, Amsterdam/Tokyo.

Breuker, J. (1997). Problems in indexing problem solving methods. In *Proceedings of the Workshop on Problem Solving Methods for Knowledge Based Systems, International Joint Conference on Artificial Intelligence (IJCAI'97)*, Nagayo, Japan.

Breuker, J. (2004). Constructing a legal core ontology: LRI-Core. In Freitas, F., Stuckenschmidt, H., and Volz, R., editors, *Proceedings WONTO-2004, Workshop on ontologies and their applications*, pages 115–126. LivroRapido, Porto Alegre, BR.

Breuker, J., Boer, A., Hoekstra, R., and van den Berg, K. (2006). Developing content for LKIF: Ontologies and frameworks for legal reasoning. In van Engers, T. M., editor, *Legal Knowledge and Information Systems. JURIX 2006: The Nineteenth Annual Conference*, volume 152 of *Frontiers in Artificial Intelligence and Applications*.

Breuker, J. and Hoekstra, R. (2004a). Core concepts of law: taking commonsense seriously. In Varzi, A. and Vieu, L., editors, *Proceedings of Formal Ontologies in Information Systems (FOIS-2004)*, pages 210–221. IOS-Press.

Breuker, J. and Hoekstra, R. (2004b). Direct: Ontology based discovery of responsibility and causality in legal case descriptions. In Gordon, T., editor, *Legal Knowledge and Information Systems. Jurix 2004: The Seventeenth Annual Conference.*, pages 59–68, Amsterdam. IOS Press.

Breuker, J. and Hoekstra, R. (2004c). Epistemology and ontology in core ontologies: FOLaw and LRI-Core, two core ontologies for law. In Gangemi, A. and Borgo, S., editors, *Proceedings of the EKAW Workshop on Core Ontologies in Ontology Engineering*. CEUR.

Breuker, J., Hoekstra, R., Boer, A., van den Berg, K., Rubino, R., Sartor, G., Palmirani, M., Wyner, A., and Bench-Capon, T. (2007). OWL ontology of basic legal concepts (LKIF-Core). Deliverable 1.4, Estrella.

Breuker, J., Valente, A., and Winkels, R. (2004). Use and reuse of legal ontologies in knowledge engineering and information management. In Benjamins, V., Casanovas, P., Breuker, J., and A.Gangemi, editors, *Law and the Semantic Web*, pages 36 – 64. Springer.

Breuker, J. and Van De Velde, W., editors (1994). *CommonKADS Library for Expertise Modeling: reusable problem solving components*. IOS-Press/Ohmsha, Amsterdam/Tokyo.

Breuker, J. and Wielinga, B. (1987). Knowledge acquisition as modelling of expertise: the KADS-methodology. In Addis, T., Boose, J., and Gaines, B., editors, *Proceedings of the European Knowledge Acquisition Workshop*, pages 102 – 110, Reading GB. Reading Press.

Breuker, J. A. (1981). *Availability of Knowledge*. PhD thesis, COWO, University of Amsterdam.

Brockmans, S., Colomb, R. M., Haase, P., Kendall, E. F., Wallace, E. K., Welty, C. A., and Xie, G. T. (2006). A model driven approach for building OWL DL and OWL full ontologies. In Cruz, I. F., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., and Aroyo, L., editors, *International Semantic Web Conference*, volume 4273 of *Lecture Notes in Computer Science*, pages 187–200. Springer.

Buchanan, B. G. and Shortliffe, E. H., editors (1984). *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*. Addison-Wesley Series in Artificial Intelligence. Addison-Wesley, Reading, MA.

Bylander, T. and Chandrasekaran, B. (1987). Generic tasks for knowledge-based reasoning: The "right" level of abstraction for knowledge acquisition. *International Journal of Man-Machine Studies*, 26(2):231–243.

Calvanese, D., de Giacomo, G., Lembo, D., Lenzerini, M., and Rosati, R. (2005). Tailoring owl for data intensive ontologies. In *Proceedings of the 1st OWL: Experiences and Directions Workshop (OWLED 2005)*.

Carbonell, J. R. (1970). *Mixed-Initiative Man-Computer Instructional Dialogues*. PhD thesis, Massachusetts Institute of Technology.

Casanovas, P., Casellas, N., Vallbe, J.-J., Poblet, M., Benjamins, R., Blazquez, M., Pena, R., and Contreras, J. (2006). Semantic web: a legal case study. In Davies, J., Studer, R., and Warren, P., editors, *Semantic Web Technologies*. Wiley.

Casellas, N., Casanovas, P., Vallbé, J.-J., Poblet, M., Blázquez, M., Contreras, J., Cobo, J. M. L., and Benjamins, V. R. (2007). Semantic enhancement for legal information retrieval: Iuriservice performance. In *The Eleventh International Conference on Artificial Intelligence and Law, Proceedings of the Conference (ICAIL 2007)*, pages 49–57, Stanford Law School, Stanford, California, USA. ACM.

Chalupsky, H. (2000). OntoMorph: a translation system for symbolic knowledge. In Cohn, A., Giunchiglia, F., and Selman, B., editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Seventh International Conference (KR2000)*, San Francisco, CA. Morgan Kaufmann.

Chandrasekaran, B. and Johnson, T. R. (1993). Generic tasks and task structures: History, critique and new directions,. In David, J. M., Krivine, J. P., and Simmons, R., editors, *Second Generation Expert Systems*. Springer Verlag.

Chaudhri, V. K., Farquhar, A., Fikes, R., Karp, P. D., and Rice, J. (1998). OKBC: A programmatic foundation for knowledge base interoperability. In *Proceedings of AAAI-98*, pages 600–607, Madison, WI.

Clancey, W. J. (1983). The epistemology of a rule-based expert system - a framework for explanation. *Artificial Intelligence*, 20(3):215–251. First published as Stanford Technical Report, November 1981.

Clark, P., Thompson, J., and Porter, B. (2000). Knowledge patterns. In Cohn, A., Giunchiglia, F., and Selman, B., editors, *Proceedings of the 7th International Conference KR'2000*, pages 591–600, CA. Kaufmann.

Collins, A. M. and Quillian, M. R. (1969). Retrieval time from semantic memory. *Journal of Verbal Learning and Verbal Behavior*, 8:240–247.

Connolly, D., van Harmelen, F., Horrocks, I., McGuinness, D. L., Patel-Schneider, P. F., and Andrea Stein, L. (2001). DAML+OIL (march 2001) reference description. Note, W3C.

Cregan, A., Schwitter, R., and Meyer, T. (2007). Sydney OWL syntax - towards a controlled natural language syntax for OWL 1.1. In Golbreich, C., Kalyanpur, A., and Parsia, B., editors, *Proceedings of OWL: Experiences and Directions (OWLED 2007)*, volume 258 of *CEUR Workshop Proceedings*, Innsbrück, Austria.

Cuenca Grau, B., Horrocks, I., Kazakov, Y., and Sattler, U. (2007). Ontology reuse: Better safe than sorry. In Calvanese, D., Franconi, E., Haarslev, V., Lembo, D., Motik, B., Turhan, A.-Y., and Tessaris, S., editors, *Description Logics*, volume 250 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Cuenca Grau, B., Motik, B., Wu, Z., Fokoue, A., and Lutz, C. (2009). OWL 2 Web Ontology Language: Profiles. Technical report, W3C.

Dahllöf, M. (1995). On the semantics of propositional attitude reports.

Davidson, D. (2001). *Essays on Actions and Events*. Oxford University Press, Oxford.

Davis, R. (1984). Diagnostic reasoning based on structure and behavior. *Artificial Intelligence*, 24:347–410.

Davis, R. and King, J. J. (1984). *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*, chapter The Origin of Rule-Based Systems in AI, pages 20–52. Addison-Wesley Series in Artificial Intelligence. Addison-Wesley, Reading, MA.

Davis, R., Shrobe, H. E., and Szlovits, P. (1993). What is knowledge representation? *AI Magazine*, 14(1):17–33.

de Maat, E., Winkels, R., and van Engers, T. (2006). Automated detection of reference structures in law. In van Engers, T. M., editor, *Legal Knowledge and Information Systems. Jurix 2006: The Nineteenth Annual Conference*, volume 152 of *Frontiers in Artificial Intelligence and Applications*, pages 41–50. IOS Press.

de Maat, E., Winkels, R., and van Engers, T. (2008). Making Sense of Legal Texts. In Grewendorf, G. and Rathert, M., editors, *Formal Linguistics and Law*, Trends in Linguistics - Studies and Monographs (TiLSM). Mouton, De Gruyter, Berlin.

Delgado, J.; Gallego, I. L. S. and García, R. (2003). IPROnto: An ontology for digital rights management. In *16th Annual Conference on Legal Knowledge and Information Systems, JURIX 2003*, volume 106 of *Frontiers in Artificial Intelligence and Applications*. IOS Press.

den Haan, N. (1992). TRACS A Support Tool for Drafting and Testing Law. In *Information Technology and Law – Fifth International Conference on Legal Knowledge Based Systems, JURIX–1992*. Koninklijke Vermande.

den Haan, N. (1996). *Automated Legal Reasoning*. PhD thesis, University of Amsterdam.

Dennett, D. C. (1987). *The Intentional Stance*. MIT-Press.

Doerr, M., Hunter, J., and Lagoze, C. (2003). Towards a core ontology for information integration. *Journal of Digital Information*, 4(1).

Donnelly, M. (2005). Relative places. *Applied Ontology*, 1(1):55–75.

Doyle, J. and Patil, R. (1991). Two theses of knowledge representation: Language restrictions, taxonomic classifications, and the utility of representation services. *Artificial Intelligence*, 48(3):261–298.

Drummond, N., Rector, A., Stevens, R., Moulton, G., Horridge, M., Wang, H., and Seidenberg, J. (2006). Putting OWL in order: Patterns for sequences in OWL. In *Proceedings of OWLED 2006*, Athens, GA.

Elhag, A. A., Breuker, J. A., and Brouwer, B. W. (1999). On the formal analysis of normative conflicts. In van den Herik et al., H., editor, *JURIX 1999: The Twelfth Annual Conference*, Frontiers in Artificial Intelligence and Applications, pages 35–46, Nijmegen. GNI.

Farquhar, A., Fikes, R., and Rice, J. (1997). The ontolingua server: a tool for collaborative ontology construction. *International Journal of Human-Computer Studies*, 46(6):707–727.

Feigenbaum, E. (1980). Knowledge engineering: the applied side of artificial intelligence. Technical report, Department of Computer Science, Stanford University.

Fensel, D., Horrocks, I., Harmelen, F. V., Decker, S., Erdmann, M., and Klein, M. (2000). OIL in a nutshell. In Dieng, R., editor, *Proceedings of the 12th European Workshop on Knowledge Acquisition, Modeling, and Management (EKAW'00)*, volume 1937 of *Lecture Notes in Artificial Intelligence*, pages 1–16. Springer-Verlag.

Fernández, M., Gómez-Pérez, A., and Juristo, N. (1997). METHONTOLOGY: From ontological art towards ontological engineering. In *AAAI-97 Spring Symposium on Ontological Engineering*, pages 33–40.

Fernández-López, M. and Gómez-Pérez, A. (2002). Overview and analysis of methodologies for building ontologies. *The Knowledge Engineering Review*, 17(2):129–156.

Forbus, K. D. (1984). Qualitative process theory. *Artificial Intelligence*, 24:85–168.

Gangemi, A. (2005). Ontology design patterns for semantic web content. In et al., Y. G., editor, *International Semantic Web Conference*, volume 3729 of *Lecture Notes in Computer Science*, pages 262–276. Springer.

Gangemi, A. and Euzenat, J., editors (2008). *Knowledge Engineering: Practice and Patterns, 16th International Conference, EKAW 2008, Acitrezza, Italy, September 29 - October 2, 2008. Proceedings*, volume 5268 of *Lecture Notes in Computer Science*. Springer.

Gangemi, A., Fisseha, F., Keizer, J., Lehmann, J., A.Liang, Pettman, I., Sini, M., and Taconet, M. (2004). A core ontology of fishery and its use in the fishery ontology service project. In Gangemi, A. and Borgo, S., editors, *Proceedings of the EKAW Workshop on Core Ontologies in Ontology Engineering*.

Gangemi, A., Guarino, N., Masolo, C., Oltramari, A., and Schneider, L. (2002). Sweetening ontologies with DOLCE. In Gomez-Perez, A. and Benjamins, V., editors, *Proceedings of the EKAW-2002*, pages 166–181. Springer.

Gangemi, A. and Mika, P. (2003). Understanding the semantic web through descriptions and situations. In *Proceedings of CoopIS/DOA/ODBASE*, pages 689–706.

Gangemi, A., Sagri, M., and Tiscornia, D. (2005). A constructive framework for legal ontologies. In Benjamins, V., Casanovas, P., Breuker, J., and Gangemi, A., editors, *Law and the Semantic Web*, pages 97–124. Springer Verlag.

Gasse, F., Sattler, U., and Haarslev, V. (2008). Rewriting rules into SROIQ axioms. In Baader, F., Lutz, C., and Motik, B., editors, *Description Logics*, volume 353 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Genesereth, M. R. and Fikes, R. E. (1992). Knowledge Interchange Format, Version 3.0 Reference Manual. Technical Report Logic-92-1, Computer Science Department, Stanford University, Stanford, CA, USA.

Genesereth, M. R. and Nilsson, N. J. (1987). *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann, Los Altos.

Georgeff, M., Pell, B., Pollack, M., Tambe, M., and Wooldridge, M. (1999). The belief-desire-intention model of agency. In Müller, J., Singh, M. P., and Rao, A. S., editors, *Proceedings of the 5th International Workshop on Intelligent Agents V : Agent Theories, Architectures, and Languages (ATAL-98)*, volume 1555, pages 1–10. Springer-Verlag: Heidelberg, Germany.

Ghilardi, S., Lutz, C., and Wolter, F. (2006). Did i damage my ontology? a case for conservative extensions in description logics. In Doherty, P., Mylopoulos, J., and Welty, C. A., editors, *KR*, pages 187–197. AAAI Press.

Gordon, T. (2007). Constructing arguments with a computational model of an argumentation scheme for legal rules. In Winkels, R., editor, *Proceedings of the Eleventh International Conference on Artificial Intelligence and Law*, pages 117–121. IAAIL, ACM.

Gordon, T., Prakken, H., and Walton, D. (2007a). The carneades model of argument and burden of proof. *Artificial Intelligence*, 171(10-11):875–896.

Gordon, T. F., van den Berg, K., Kordelaar, P., Lee, J., Sekkat, S., and Wyner, A. (2007b). Formal specifications of the knowledge formats of participating LKBS vendors. Deliverable 1.2, Estrella.

Grenon, P. (2003). Nuts in BFO's nutshell: Revisions to the bi-categorial axiomatization of BFO. Technical Report ISSN 1611-4019, Institute for Formal Ontology and Medical Information Science (IFOMIS).

Grosof, B., Volz, R., Horrocks, I., and Decker, S. (2003). Description logic programs: Combining logic programs with description logics. In *Proceedings of the 12th International World Wide Web Conference (WWW 2003)*.

Grossi, D. (2007). *Desigining Invisible Hancuffs. Formal Investigations in Institutions and Organizations for Multi-agent Systems*. SIKS dissertation series, Utrecht University. 2007-16.

Grossi, D., Meyer, J.-J. C., and Dignum, F. (2005). Modal logic investigations in the semantics of counts-as. In Gardner, A., editor, *Proceedings of the Tenth International Conference on Artificial Intelligence and Law (ICAIL)*, pages 125–140, Bologna, Italy. IAAIL, ACM Press.

Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220.

Gruber, T. R. (1994). Toward principles for the design of ontologies used for knowledge sharing. In Guarino, N. and Poli, R., editors, *Formal Ontology in Conceptual Analysis and Knowledge Representation*, Deventer, The Netherlands. Kluwer Academic Publishers.

Grüninger, M. and Fox, M. S. (1995). Methodology for the design and evaluation of ontologies. In *IJCAI'95, Workshop on Basic Ontological Issues in Knowledge Sharing*.

Guarino, N. (1994). The ontological level. In Casati, R., Smith, B., and White, G., editors, *Philosophy and the Cognitive Sciences, Proceedings of the 16th International Wittgenstein Symposium*, pages 443–456, Vienna. Hölder-Pichler-Tempsky.

Guarino, N. (1997). Understanding, building, and using ontologies: A commentary to using explicit ontologies in kbs development, by van heijst, schreiber, and wielinga. *International Journal of Human and Computer Studies*, 46:293–310.

Guarino, N. (1998). Formal ontology and information systems. In Guarino, N., editor, *Proceedings of FOIS'98*, pages 3–15, Trento, Italy. IOS Press.

Guarino, N. and Giaretta, P. (1995). Ontologies and knowledge bases: Towards a terminological clarification. In Mars, N., editor, *Towards Very Large Knowledge Bases – Knowledge Building and Knowledge Sharing 1995*, pages 25–32, Amsterdam. IOS Press.

Guarino, N. and Welty, C. (2002). Evaluating ontological decisions with OntoClean. *Communications of the ACM*, 45(2):61–65.

Guarino, N. and Welty, C. A. (2004). An Overview of OntoClean. In Staab, S. and Studer, R., editors, *Handbook on Ontologies*, chapter 8. Springer Verlag.

Halaschek-Wiener, C., Katz, Y., and Parsia, B. (2006). Belief base revision for expressive description logics. In *OWL: Experiences and Directions 2006*.

Hamscher, W. C., Console, L., and de Kleer, J., editors (1992). *Readings in Model-Based Diagnosis*. Morgan Kaufmann, San Mateo, CA.

Hart, H. and Honoré, T. (1985). *Causation in the Law*. Oxford University Press, New York, second edition edition.

Haslanger, S. (2003). *The Oxford Handbook of Metaphysics*, chapter Persistence Through Time, pages 315–354. Oxford University Press.

Hayes, P. (1977). In defence of logic. In Reddy, R., editor, *Proceedings of the 5th International Joint conference on Artificial Intelligence (IJCAI 1977)*, pages 559–565, Cambridge, MA. William Kaufmann.

Hayes, P. (2004). RDF semantics. Recommendation, W3C.

Hayes, P. J. (1985). The second naive physics manifesto. In Hobbs, J. R. and Moore, R. C., editors, *Formal Theories of the Commonsense World*, volume 1 of *Ablex series in Artificial Intelligence*, pages 1–36. Ablex, Norwood, NJ.

Heflin, J., Hendler, J., and Luke, S. (1999). SHOE: A knowledge representation language for internet applications. Technical Report CS-TR-4078 (UMIACS TR-99-71), Dept. of Computer Science, University of Maryland at College Park.

Herre, H., Heller, B., Burek, P., Hoehndorf, R., Loebe, F., and Michalek, H. (2006). General formal ontology (GFO) - basic principles. Onto-med report, Institute of Medial Informatics, Statistics and Epidemiology (IMISE).

Hobbs, J. R. (1995). Sketch of an ontology underlying the way we talk about the world. *International Journal of Human-Computer Studies*, 43:819–830.

Hoekstra, R. (2001). Errors in modelling and representation. Master's thesis, University of Amsterdam.

Hoekstra, R. (2008). Use of OWL in the legal domain (statement of interest). In Clark, K. and Patel-Schneider, P. F., editors, *Proceedings of OWL: Experiences and Directions (OWLED 2008 DC)*, Washington, DC (metro).

Hoekstra, R. and Breuker, J. (2007). Commonsense causal explanation in a legal domain. *Artificial Intelligence & Law*, 15(3):281–299. Special issue on Legal Ontologies and Artificial Intelligence Techniques.

Hoekstra, R. and Breuker, J. (2008). Polishing diamonds in OWL 2. In Gangemi, A. and Euzenat, J., editors, *Proceedings of the 16th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2008)*, LNAI/LNCS. Springer Verlag. To be published.

Hoekstra, R., Breuker, J., Bello, M. D., and Boer, A. (2007). The LKIF Core ontology of basic legal concepts. In Casanovas, P., Biasiotti, M. A., Francesconi, E., and Sagri, M. T., editors, *Proceedings of the Workshop on Legal Ontologies and Artificial Intelligence Techniques (LOAIT 2007)*.

Hoekstra, R., Breuker, J., Di Bello, M., and Boer, A. (2008). LKIF Core: Principled ontology development for the legal domain. In Breuker, J., Casanovas, P., Klein, M., and Francesconi, E., editors, *Law, Ontologies and the Semantic Web*, Frontiers of Artificial Intelligence and Applications. IOS Press, Amsterdam. To be Published.

Hoekstra, R., Liem, J., Bredeweg, B., and Breuker, J. (2006). Requirements for representing situations. In Grau, B. C., Hitzler, P., Shankey, C., and Wallace, E., editors, *Proceedings of the OWLED'06 workshop on OWL: Experiences and Directions 2006*, volume 216 of *CEUR Workshop Proceedings*, Athens, Georgia (USA).

Hohfeld, W. (1919). *Fundamental Legal Conceptions as Applied in Legal Reasoning*. Yale University Press. Edited by W.W. Cook, fourth printing, 1966.

Horn, W., editor (1990). *Causal AI Models: Steps Toward Applications*. Hemisphere Publishing Corporation.

Horridge, M., Jupp, S., Moulton, G., Rector, A., Stevens, R., and Wroe, C. (2007). A practical guide to building OWL ontologies using Protégé 4 and CO-ODE tools. Technical Report 1.1, The University of Manchester.

Horrocks, I. (2000). A denotational semantics for standard OIL and instance OIL. Technical report, OnTo Knowledge Project.

Horrocks, I., Kutz, O., and Sattler, U. (2006). The even more irresistible $\mathcal{SROIQ}$. In *Proceedings of the Tenth International Conference on Principles of Knowledge Representation and Reasoning*, pages 57–67.

Horrocks, I. and Patel-Schneider, P. F. (2003). Three theses of representation in the semantic web. In *Proceedings of the 12th International Conference on World Wide Web (WWW 2003)*, pages 39 – 47, Budapest, Hungary. ACM Press.

Horrocks, I., Patel-Schneider, P. F., and van Harmelen, F. (2003). From $\mathcal{SHIQ}$ and RDF to OWL: The making of a web ontology language. *Journal of Web Semantics*, 1(1):7–26.

Horrocks, I. and Sattler, U. (2001). Ontology reasoning in the $\mathcal{SHOQ}(D)$ description logic. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*.

Horrocks, I., Sattler, U., and Topies, S. (1999). Practical reasoning for expressive description logics. In Ganzinger, H., McAllester, D., and Voronkov, A., editors, *Proceedings of the 6th International Conference on Logic for Programming and Automated Reasoning (LPAR'99)*, volume 1705 of *Lecture Notes in Artificial Intelligence*, pages 161 – 180. Springer Verlag.

ISO/IEC (2007). Common Logic (CL) - a framework for a family of logic-based languages. Technical Report 24707:2007, ISO/IEC.

Jansweijer, W., Breuker, J., van Lieshout, J., van der Stadt, E., Hoekstra, R., and Boer, A. (2001). Workflow Directed Knowledge Management. In Stanford-Smith, B. and Chiozza, E., editors, *E-work and E-commerce: Novel solutions and practices for a global networked economy*, pages 755–762, Amsterdam. IOS-Press. ISBN 1.58603.205.4.

Jimenez-Ruiz, E., Cuenca Grau, B., Sattler, U., Schneider, T., and Berlanga, R. (2008). Safe and economic re-use of ontologies: A logic-based methodology and tool support. In *Proceedings of OWLED 2008 DC*.

Kaljurand, K. and Fuchs, N. E. (2007). Verbalizing OWL in attempto controlled english. In Golbreich, C., Kalyanpur, A., and Parsia, B., editors, *Proceedings of OWL: Experiences and Directions (OWLED 2007)*, volume 258 of *CEUR Workshop Proceedings*, Innsbrück, Austria.

Kant, I. (1997). *Lectures on Metaphysics. Part III: Metaphysic Mrongovius (1782-1783)*. Cambridge University Press, Cambridge.

Kelsen, H. (1991). *General Theory of Norms*. Clarendon Press, Oxford.

Kim, J. (1998). *Mind in a Physical World: An Essay on the Mind-Body Problem and Mental Causation*. Representation and Mind. MIT Press, Cambridge, Massachusetts. Series editors: Hilary Putnam and Ned Block.

Klarman, S., Hoekstra, R., and Bron, M. (2008). Versions and applicability of concept definitions in legal ontologies. In Clark, K. and Patel-Schneider, P. F., editors, *Proceedings of OWL: Experiences and Directions (OWLED 2008 DC)*, Washington, DC (metro).

Klein, M. (2001). Combining and relating ontologies: An analysis of problems and solutions. In *Proceedings of the Workshop on Ontologies and Information Sharing (at IJCAI 2001)*, pages 53–62, Seattle, WA.

Konev, B., Lutz, C., Walther, D., and Wolter, F. (2008). CEX and MEX: Logical diff and semantic module extraction in a fragment of OWL. In *Proceedings of OWLED 2008 DC*.

Kowalski, R. A. (1974). Predicate logic as a programming language. In *Proceedings of the IFIP Congress*, pages 569–574, Stockholm. North Holland Publishing Co.

Krötzsch, M., Rudolph, S., and Hitzler., P. (2006). On the complexity of horn description logics. In Cuenca Grau, B., Hitzler, P., Shankey, C., and Wallace, E., editors, *Proceedings of the 2nd Workshop on OWL: Experiences and Directions (OWLED-06)*. CEUR Workshop Proceedings.

Krötzsch, M., Rudolph, S., and Hitzler, P. (2008a). Description logic rules. In Ghallab, M., Spyropoulos, C. D., Fakotakis, N., and Avouris, N., editors, *Proceedings of the 18th European Conference on Artificial Intelligence (ECAI-08)*, pages 80–84. IOS Press.

Krötzsch, M., Rudolph, S., and Hitzler, P. (2008b). ELP: Tractable rules for OWL 2. Technical report, Institute AIFB, Universität Karlsruhe, Karlsruhe, Germany.

Krötzsch, M., Rudolph, S., and Hitzler, P. (2008c). Expressive tractable description logics based on $\mathcal{SROIQ}$ rules. Technical report, Institut AIFB, Universität Karlsruhe (TH), Karlsruhe, Germany.

Lakoff, G. (1987). *Women, Fire and Dangerous Things*. University of Chicago Press.

Lakoff, G. and Núñez, R. (2000). *Where Mathematics Comes From*. Basic Books.

Lame, G. (2004). Using nlp techniques to identify legal ontology components: Concepts and relations. *Artificial Intelligence and Law, This Issue?*

Lehmann, J. (2003). *Causation in Artificial Intelligence and Law - A modelling approach*. PhD thesis, University of Amsterdam - Faculty of Law - Department of Computer Science and Law.

Lehmann, J., Borgo, S., Masolo, C., and Gangemi, A. (2004). Causality and causation in DOLCE. In Varzi, A. C. and Vieu, L., editors, *Formal Ontology in Information Systems, Proceedings of the International Conference FOIS 2004*, Frontiers in Artificial Intelligence and Applications, pages 273–284, Torino. IOS Press.

Lehmann, J. and Gangemi, A. (2007). An ontology of physical causation as a basis for assessing causation in fact and attributing legal responsibility. *Artificial Inteligence and Law*. Special Issue on Legal Ontologies and Artificial Intelligence Techniques.

Leibniz, G. W. (1903). *Opuscules et fragments inédits de Leibniz: extraits des manuscrits de la Bibliothèque royale de Hanovre*. Alcan, Paris. Reprinted Hildesheim, Georg Olms, 1961.

Lenat, D. B. (1995). CYC: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11):33–38.

Lenat, D. B., Guha, R. V., Pittman, K., Pratt, D., and Shepherd, M. (1990). Cyc: toward programs with common sense. *Commun. ACM*, 33(8):30–49.

Levesque, H. and Brachman, R. (1985). A fundamental tradeoff in knowledge representation and reasoning. In Levesque, H. and Brachman, R., editors, *Readings in Knowledge Representation*, pages 41–70. Morgan Kaufmann.

Levesque, H. J. (1984). Foundations of a functional approach to knowledge representation. *Artificial Intelligence*, 23:155–212.

Levesque, H. J. and Brachman, R. J. (1987). Expressiveness and tracability in knowledge representation and reasoning. *Computational Intelligence*, 3:78–93.

Loebe, F. (2003). An analysis of roles. Technical Report 6, Research Group Ontologies in Medicine, University of Leipzig. Onto-Med Report.

Loebe, F. (2007). Abstract vs. social roles, towards a general theoretical account of roles. *Applied Ontology*, 2(2):127–158.

Lutz, C. (1999). The complexity of reasoning with concrete domains. LTCS Report 99-01, RWTH Aachen. Preliminary Version.

MacGregor, R. and Bates, R. (1987). The LOOM knowledge representation language. Technical Report ISI/RS-87-188, ISI, University of Southern California.

Marcus, S., editor (1988). *Automating Knowledge Acquisition for Expert Systems*. Kluwer, Reading MA.

Masolo, C., Borgo, S., Gangemi, A., Guarino, N., and Oltramari, A. (2003). WonderWeb ontology library. Deliverable D18, version 1, ISTC-CNR (Italy).

Masolo, C., Vieu, L., Bottazzi, E., Catenacci, C., Ferrario, R., Gangemi, A., and Guarino, N. (2004). Social roles and their descriptions. In *Proceedings of Knowledge Representation Workshop*.

McCarthy, J. (1959). Programs with common sense. In *Proceedings of the Teddington Conference on the Mechanization of Thought Processes*, pages 75–91, London. Her Majesty's Stationary Office.

McCarthy, J. (1980). Circumscription – a form of non-monotonic reasoning. *Artificial Intelligence*, 13:27–39.

McCarthy, J. and Hayes, P. J. (1969). Some philosophical problems from the standpoint of artificial intelligence. *Machine Intelligence*, 4.

McGuinness, D. and van Harmelen, F. (2004). OWL web ontology language overview. W3C recommendation, World Wide Web Consortium (W3C).

Michon, J. (1990). Implicit and explicit representations of time. In Block, R., editor, *Cognitive models of psychological time*, pages 37–58. Lawrence Erlbaum.

Miles, A. and Bechhofer, S. (2008). SKOS simple knowledge organization system reference. Working draft, W3C.

Miller, G. (1990). WordNet: An on-line lexical database. *International Journal of Lexicography*, 3(4). (Special Issue).

Minsky, M. (1975). A framework for representing knowledge. In Winston, P., editor, *The Psychology of Computer Vision*. McGraw-Hill, New York.

Minsky, M. (1982). Why people think computers can't. *AI Magazine*, 3(4).

Minsky, M. (1984). *The Society of Mind*. Simon & Schuster Inc.

Mommers, L. (2002). *Applied legal epistemology: building a knowledge-based ontology of the legal domain*. PhD thesis, University of Leiden.

Moser, M. G. (1983). An overview of nikl, the new implementation of klone. Technical Report 5421, Bolt, Beranek, and Newman, Inc., Cambridge, MA. Research in Natural Language Understanding.

Motik, B. (2007). On the properties of metamodeling in OWL. *Journal of Logic and Computation*, 17(4):617–637.

Motik, B., Cuenca Grau, B., and Sattler, U. (2007a). Structured objects in OWL: Representation and reasoning. Technical report, University of Oxford, UK.

Motik, B., Horrocks, I., and Sattler, U. (2007b). Bridging the gap between OWL and relational databases. In Patel-Schneider, P. and Shenoy, P., editors, *Proceedings of the 16th International Conference on World Wide Web (WWW 2007)*, Banff, Alberta, Canada. ACM Press.

Motik, B., Patel-Schneider, P. F., Parsia, B., Bock, C., Fokoue, A., Haase, P., Hoekstra, R., Horrocks, I., Ruttenberg, A., Sattler, U., and Smith, M. (2009). OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax. Technical report, W3C.

Motik, B., Sattler, U., and Studer, R. (2005). Query answering for OWL-DL with rules. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, 3(1):41 – 60.

Motta, E. (1999). *Reusable Components for Knowledge Modelling*. FAIA-Series. IOS Pres, Amsterdam NL.

Mylopoulos, J. (1981). An overview of knowledge representation. In Brodie, M. L. and ZIlles, S. N., editors, *Proceedings of the Workshop on Data Abstraction, Databases and Conceptual Modelling, Pingree Park, Colorado, June 23-26, 1980*, volume 11 of *SIGMOD Record*, pages 5–12. ACM.

Neches, R., Fikes, R., Finin, T., Gruber, T., Paqtil, R., Senator, T., and Swartout, W. R. (1991). Enabling technology for knowledge sharing. *AI Magazine*, 12(3):36–56.

Nejdl, W., Wolpers, M., and Capelle, C. (2000). The RDF schema specification revisited. In *Modelle und Modellierungssprachen in Informatik un Wirtschaftsinformatik, Proceedings of Modellierung 2000*.

Newell, A. (1982). The knowledge level. *Artificial Intelligence*, 18(1):87–127.

Newell, A. (1993). Reflections on the knowledge level. *Artificial Intelligence*, 59:31–38.

Newell, A. and Simon, H. A. (1972). *Human Problem Solving*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

Niles, I. and Pease, A. (2001). Towards a standard upper ontology. In Welty, C. and Smith, B., editors, *Proceedings of the 2nd International Conference on Formal Ontology in Information Systems (FOIS-2001)*, Ogunquit, Maine.

Noy, N. (2005). Representing classes as property values on the semantic web. Working group note, W3C. http://www.w3.org/TR/swbp-classes-as-values/.

Noy, N. and Rector, A. (2006). Defining n-ary relations on the semantic web. Working group note, W3C. http://www.w3.org/TR/swbp-n-aryRelations/.

Noy, N. F. and Musen, M. A. (2000). PROMPT: Algorithm and tool for automated ontology merging and alignment. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 450 – 455.

Nuyts, J., Byloo, P., and Diepeveen, J. (2005). On deontic modality, directivity, and mood.

Oberle, D., Ankolekar, A., and et al., P. H. (2007). DOLCE ergo SUMO: On foundational and domain models in the SmartWeb Integrated Ontology (SWIntO). *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, 5:156–174.

Ogden, C. K. (1930). *Basic English: A General Introduction with Rules and Grammar*.

Orwell, G. (1949). *Nineteen Eighty-Four. A novel*.

Pan, J. and Horrocks, I. (2002). Metamodelling architecture of web ontology languages. In Gruz, I., Decker, S., Euzenat, J., and McGuinness, D., editors, *The Emerging Semantic Web*, Frontiers in Artificial Intelligence and Applications. IOS Press.

Pan, J. Z. and Horrocks, I. (2003). RDFS(FA) and RDF MT: Two Semantics for RDFS. In Fensel, D., Sycara, K., and Mylopoulos, J., editors, *Proceedings of the 2nd International Semantic Web Conference (ISWC2003)*.

Parsia, B., Halaschek-Wiener, C., and Sirin, E. (2006). Towards incremental reasoning through updates in owl-dl. In *Proceedings of WWW 2006*, Edinburgh, UK.

Patel-Schneider, P. (1984). Small can be beautiful in knowledge representation. In *Proceedings of IEEE workshop on principles of knowledge-based systems*.

Patel-Schneider, P. F., Hayes, P., and Horrocks, I. (2004). OWL web ontology language semantics and abstract syntax. Recommendation, World Wide Web Consortium (W3C).

Patel-Schneider, P. F. and Swartout, B. (1993). Description-Logic Knowledge Representation System Specification from the KRSS Group. Technical report, Bell-Labs Research.

Pearl, J. (2000). *Causality: Models, Reasoning and Inference*. Cambridge University Press, Cambridge.

Pedrinaci, C., Domingue, J., and de Medeiros, A. K. A. (2008). A core ontology for business process analysis. In Bechhofer, S. and Hauswirth, M., editors, *Proceedings of the 5th European Semantic Web Conference (ESWC 2008)*, Tenerife, Spain.

Pepper, S., Vitali, F., Marius, L., Gessa, N., and Presutti, V. (2006). A survey of RDF/Topic Maps interoperability proposals. Working group note, W3C.

Pinker, S. (2007). *The Stuff of Thought*. Allen Lane, Penguin Books, London, England.

Pinto, H. S. and Martins, J. P. (2000). Reusing ontologies. In *Proceedings of AAAI2000 Spring Symposium Series, Workshop Bringing Knowledge to Business Processes*, number SS-00-03 in AAAI Technical Report, pages 77–84, Menlo Park, California, USA. AAAI Press.

Post, E. L. (1943). Formal reductions of the general combinatorial decision problem. *American Journal of Mathematics*, 65(2):197–215.

Presutti, V., Gangemi, A., David, S., de Cea, G. A., Suárez-Figueroa, M. C., Ponsoda, E. M., and Poveda, M. (2008). A library of ontology design patterns: reusable solutions for collaborative design of networked ontologies. Deliverable D2.5.1, NeOn Project.

Puerta, A. R., Egar, J. W., Tu, S. W., and Musen, M. A. (1992). A multiple-method shell for the automatic generation of knowwledge acquisition tools. *Knowledge Acquisition*, 4:171–196.

Quillian, M. R. (1966). Semantic memory. Report AFCRL-66-189, Bolt Beranek & Newman, Cambridge, Massachusetts.

Rector, A. (2005). Representing specified values in OWL: "value partitions" and "value sets". Working group note, W3C. http://www.w3.org/TR/swbp-specified-values/.

Rector, A., Drummond, N., Horridge, M., Rogers, J., Knublauch, H., Stevens, R., Wang, H., and Wroe, C. (2004). OWL pizzas: Practical experience of teaching OWL-DL: Common errors & common patterns. In Motta, E., Shadbolt, N., Stutt, A., and Gibbins, N., editors, *Proceedings of the European Conference on Knowledge Acquisition*, pages 63–81.

Rector, A. L. (2003). Modularisation of domain ontologies implemented in description logics and related formalisms including owl. In *Proceedings of K-CAP'03*, pages 121 – 128, Sanibel Island, Florida. ACM Press.

Rubino, R., Rotolo, A., and Sartor, G. (2006). An owl ontology of fundamental legal concepts. In van Engers, T. M., editor, *Legal Knowledge and Information Systems. JURIX 2006: The Nineteenth Annual Conference*, volume 152 of *Frontiers of Artificial Intelligence and Applications*. IOS Press.

Sartor, G. (2006). Fundamental legal concepts: A formal and teleological characterisation. Technical report, European University Institute, Florence / Cirsfid, University of Bologna.

Schaffer, J. (2003). The metaphysics of causation. Stanford Encyclopedia of Philosophy.

Schank, R. and Abelson, R. (1975). Scripts, plans, and knowledge. In *Proceedings of the 4th International Joint Conference on Artificial Intelligence*, Tblisi, USSR.

Schank, R. and Abelson, R. (1977). *Scripts, Plans Goals and Understanding*. Lawrence Erlbaum, New Jersey.

Schreiber, G., Akkermans, H., Anjewierden, A., de Hoog, R., Shadbolt, N., Van den Velde, W., and Wielinga, B. (2000). *Knowledge Engineering and Management: The CommonKADS Methodology*. MIT Press.

Schreiber, G., Wielinga, B., Akkermans, H., van de Velde, W., and Anjewierden, A. (1994). Cml: The CommonKADS conceptual modelling language. In *A Future for Knowledge Acquisition*, volume 867 of *LNCS*, pages 1–25. Springer Verlag.

Schreiber, G., Wielinga, B., and Jansweijer, W. (1995). The kaktus view on the 'o' world. In *Proceedings of IJCAI95 Workshop on Basic Ontological Issues in Knowledge Sharing*, Montreal, Canada.

Schwitter, R., Kaljurand, K., Cregan, A., Dolbear, C., and Hart, G. (2008). A comparison of three controlled natural languages for OWL 1.1. In Clark, K. G. and Patel-Schneider, P. F., editors, *Proceedings of OWL: Experiences and Directions (OWLED 2008 DC)*, Washington, DC (metro).

Searle, J. R. (1995). *The Construction of Social Reality*. The Free Press, New Yor, NY.

Seidenberg, J. and Rector, A. L. (2006). Representing transitive propagation in OWL. In Embley, D. W., Olivé, A., and Ram, S., editors, *ER*, volume 4215 of *Lecture Notes in Computer Science*, pages 255–266. Springer.

Smith, B. (1978). An essay in formal ontology. *Grazer Philosophische Studien*, 6:39–62.

Smith, B. (2004). Beyond concepts: Ontology as reality representation. In Varzi, A. C. and Vieu, L., editors, *Formal Ontologies in Information Systems, Proceedings of the Third International Conference (FOIS-2004)*. IOS Press.

Smith, B. (2005). Against fantology. In Reicher, M. and Marek, J., editors, *Experience and Analysis*, pages 153–170. öbvahpt, Wien.

Smith, M. K., Welty, C., and McGuinness, D. L. (2004). OWL web ontology language guide. W3C recommendation, World Wide Web Consortium (W3C).

Sowa, J. F. (2000). *Knowledge Representation: Logical Philosophical, and Computational Foundations*. Brooks Cole Publishing Co, Pacific Grove, CA.

Stefik, M. (1986). The next knowledge medium. *AI Magazine*, 7(1):34–46.

Steimann, F. (2000). On the representation of roles in object-oriented and conceptual modelling. *Data and Knowledge Engineering*, 35(1):83–106.

Swartout, B., Patil, R., Knight, K., and Russ, T. (1996). Toward distributed use of large-scale ontologies. In *Proceedings of the Knowledge Acquisition Workshop (KAW'96)*, Banff, Alberta, Canada.

Uschold, M. (1996). Building ontologies: Towards a unified methodology. In *16th Annual Conf. of the British Computer Society Specialist Group on Expert Systems*, Cambridge, UK.

Uschold, M. and Grüninger, M. (1996). Ontologies: principles, methods, and applications. *Knowledge Engineering Review*, 11(2):93–155.

Uschold, M. and King, M. (1995). Towards a methodology for building ontologies. In *Workshop on Basic Ontological Issues in Knowledge Sharing, IJCAI-95*, Montreal, Canada.

Valente, A. (1995). *Legal Knowledge Engineering: A Modelling Approach*. PhD thesis, University of Amsterdam, Amsterdam.

Valente, A. (2005). Types and roles of legal ontologies. In Benjamins, V., Casanovas, P., Breuker, J., and Gangemi, A., editors, *Law and the Semantic Web*, volume 3369 of *Lecture Notes in Artificial Intelligence*, pages 65–76. Springer.

Valente, A. and Breuker, J. (1995). ON-LINE: An architecture for modelling legal information. In *International Conference on Artificial Intelligence and Law (ICAIL-1995)*, pages 307–315.

Valente, A. and Breuker, J. (1996). Towards principled core ontologies. In Gaines, B. and Musen, M., editors, *Proceedings of the Knowledge Acquisition Workshop - 96*, Banff, Canada.

Valente, A. and Breuker, J. (1999). Legal modelling and automated reasoning with ON-LINE. *International Journal of Human-Computer Studies*, 51(6):1079–1125.

Valente, A., Breuker, J., and van de Velde, W. (1998). The *Common*KADS library in perspective. *International Journal of Human–Computer Studies*, 49:391–416.

van de Ven, S., Breuker, J., Hoekstra, R., Wortel, L., and El-Ali, A. (2008a). Automated legal assessment in OWL 2. In *Legal Knowledge and Information Systems. Jurix 2008: The 21st Annual Conference*, Frontiers in Artificial Intelligence and Applications. IOS Press.

van de Ven, S., Hoekstra, R., Breuker, J., Wortel, L., and El-Ali, A. (2008b). Judging Amy: Automated legal assessment using OWL 2. In *Proceedings of OWL: Experiences and Directions (OWLED 2008 EU)*.

van Engers, T. and Glassée, E. (2001). Facilitating the legislation process using a shared conceptual model. *IEEE Intelligent Systems*, 16(1):50–58.

van Engers, T., Kordelaar, P., Hartog, J. D., and Glassée, E. (2000). POWER: Programme for an ontology based working environment for modeling and use of regulations and legislation. In Tjoa, Wagner, and Al-Zobaidie, editors, *In Proceedings of the 11th workshop on Databases and Expert Systems Applications (IEEE)*, pages 327–334, Greenwich London. ISBN: 0-7695-0680-1.

van Harmelen, F. and Balder, J. (1992). $(ML)^2$: a formal language for KADS models of expertise. *Knowledge Acquisition*, 4(1):127–161. Special issue: 'The KADS approach to knowledge engineering'.

van Heijst, G., Schreiber, A. T., and Wielinga, B. (1997). Using explicit ontologies for kbs development. *International Journal of Human-Computer Studies*, 46(2/3):183–292.

van Lambalgen, M. and Hamm, F. (2005). *The Proper Treatment of Events*, volume 4 of *Explorations in Semantics*. Blackwell Publishing.

Vilain, M. (1984). Kl-two, a hybrid knowledge representation system. Technical report, BBN Laboratories, Cambridge, MA.

Villanueva-Rosales, N. and Dumontier, M. (2007). Describing chemical functional groups in owl-dl for the classification of chemical compounds. In *OWL: Experiences and Directions (OWLED 2007)*.

Visser, P. R., Jones, D. M., Bench-Capon, T., and Shave, M. (1997). An analysis of ontology mismatches; heterogeneity versus interoperability. In *AAAI 1997 Spring Symposium on Ontological Engineering*, Stanford, CA.

Vrandečić, D. (2005). Explicit knowledge engineering patterns with macros. In Welty, C. and Gangemi, A., editors, *Proceedings of the Ontology Patterns for the Semantic Web Workshop at the ISWC 2005*, Galway, Ireland.

Wang, T. D. and Parsia, B. (2007). Ontology performance profiling and model examination: First steps. In et al., K. A., editor, *ISWC/ASWC*, volume 4825 of *Lecture Notes in Computer Science*, pages 595–608. Springer.

Wielinga, B. and Schreiber, A. (1993). Reusable and sharable knowledge bases: a european perspective. In *Proceedings of First International Conference on Building and Sharing of Very Large-Scaled Knowledge Bases*, Tokyo, Japan. Information Processing Development Center.

Wielinga, B. J., Schreiber, A. T., and Breuker, J. A. (1992). KADS: A modelling approach to knowledge engineering. *Knowledge Acquisition*, 4:5–53.

Winkels, R., Boer, A., Breuker, J., and Bosscher, D. (1998). Assessment based legal information serving and cooperative dialogue in clime. In *Proceedings of Jurix-98*, pages 131–146, Nijmegen, Netherlands. GNI.

Winkels, R., Boer, A., and Hoekstra, R. (2002). CLIME: lessons learned in legal information serving. In van Harmelen, F., editor, *Proceedings of the European Conference on Artificial Intelligence (ECAI 2002)*, Amsterdam. IOS-Press.

Winkels, R., Bosscher, D., Boer, A., and Breuker, J. (1999). Generating exception structures for legal information serving. In Gordon, T., editor, *Proceedings of the Seventh International Conference on Artificial Intelligence and Law (ICAIL-99)*, pages 182–195. ACM, New York.

Winkels, R. and de Bruijn, H. (1996). Case frames: Bridging the gap between a case and the law. In Carr, I. and Narayanan, A., editors, *Proceedings of the First European Conference on Computers, Law and AI*, pages 205–213, Exeter. EUCLID.

Woods, W. (1975). What's in a link? foundations for semantic networks. In Bobrow, D. and Collins, A., editors, *Representation and Understanding*, pages 35–82. Academic Press, New York.

# Index