Ontology Representation : design patterns and ontologies that make sense

Hoekstra, R.J.

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 favourable 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.