Ontology Representation : design patterns and ontologies that make sense

Hoekstra, R.J.

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.