Representing Social Reality in OWL 2[⋆]

Rinke Hoekstra[1,2]

[1] Department of Computer Science, Faculty of Exact Sciences, VU University Amsterdam
`hoekstra@few.vu.nl`
[2] Leibniz Center for Law, Faculty of Law, University of Amsterdam
`hoekstra@uva.nl`

Abstract This paper introduces a design pattern that allows for the OWL 2 DL representation of concepts central to social reality: roles. The work presented here is motivated by experiences in the development of the LKIF Core ontology of basic legal concepts [10,8]. This paper applies modelling steps identified in earlier work for the representation of transactions [9] to the domain of roles. This is done by building on Searle's theory of social reality [14]. We use the new features of OWL 2 to approximate a reified relation, and show how the approach of [9] can be reused to define a pattern for capturing roles, intentional concepts and n-ary relations [13].

1 Introduction

This paper introduces a design pattern that allows for the OWL 2 DL representation of relational roles in social reality, and n-ary relations in general. We apply the pattern to two use cases: the representation of *n-ary relations* by the SWBP of the W3C[3] [13] and a description of *social reality* [14].

The work presented here is motivated by experiences in the development of the LKIF Core ontology of basic legal concepts [10,8]. To be able to adequately capture the contents of legal norms [17], this ontology had to be equipped with a large number of concepts for describing social reality: roles, beliefs, desires, obligations, permissions, intentions etc. The problem of representing roles such as 'student' has been discussed at length in the literature [15,12]: they can be seen both as a class ('Jane is a Student') and as a relation (e.g. the student relation between Jane and her university). The same duality holds for intentional categories such as beliefs and desires. While a class conveys a stronger ontological commitment than the relation, the relation is often more convenient and succinct for practical use.

The representational construct in existing Semantic Web languages that comes closest to capturing this duality is reification. Reification allows one to address the relation between two or more resources (e.g. an RDF triple) as a

[⋆] This paper is a significantly revised version of a section in [8]

[3] SWBP: Semantic Web Best Practices and Deployment group. See `http://www.w3.org/2001/sw/BestPractices`.

primary entity (a resource). It is a sensitive subject, and is often associated with messy modelling and has complicated semantics, especially when different levels of reification are not stratified. On the other hand, reification is becoming increasingly important as a candidate approach to attaching metadata to individual triples, e.g. for the purposes of provenance tracking.

Unfortunately, the built-in reification capabilities of RDF are not sufficient to our purposes. Namely, the existence of a reification (an RDF statement) does not entail the existence of the corresponding relation [6]. Furthermore, OWL 2 DL [2] is not expressive enough to represent reified relations either, even though these form a core part of use cases in social reality.

Current Semantic Web languages fall short in another, related respect: they do not provide means to express n-ary relations. This issue was addressed by the SWBP in the identification of a (famous) design pattern that represents the n-ary relation as a class with multiple properties relating it to the relata of the n-ary relation [13]. Unavoidably, this pattern has an effect similar to that of RDF reification: the original relation between the relata is lost.

This paper applies modelling steps identified in earlier work for the representation of transactions [9] to the domain of roles. This is done by building on Searle's theory of social reality [14]. We use the complex role inclusion axioms of OWL 2 to approximate a reified relation. We show how the new features of OWL 2 can be used to define a pattern for capturing intentional concepts, speech acts, and the n-ary relations examples of the SWBP document [13].

Section 2 describes the problems of representing social roles and n-ary relations in more detail. section 3 applies the steps of [9] to construct the design pattern, and subsection 3.1 applies this pattern to the use cases described above.

2 The Problem

The LKIF Core ontology distinguishes three levels: physical reality, an intentional level and a legal level. Each consecutive level builds on and expands the level below it with more descriptive power. The distinction between these levels is inspired by Valente's functional ontology, where legal knowledge is considered to be an abstraction of commonsense knowledge [16].

Categories described at the intentional level and above, are *social constructs* that can be attributed to, or imposed on brute facts [14]. Brute facts are phenomena of which the existence does not depend on human agreement. This is similar to the way in which intentional and functional notions are generalised over physical phenomena in the design and intentional stance of Dennett [1]. According to Searle, institutional facts are constructed by means of *constitutive* and *regulative* rules. These are 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).

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. Because of the normative character of regulative rules, the *counts-as* relation has received a lot of attention in AI and Law, most recently in [4].

Constitutive rules are deceptively similar to subsumption, but are very distinct in three ways. Firstly, 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. Secondly, 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. And thirdly, the counts-as relation can be used to connect *anti-rigid* with *rigid* classes [5]. For example, where "bills issued by the Bureau of Engraving and Printing" is rigid, "money" 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. In the following section we briefly discuss existing approaches to this problem.

2.1 Existing Approaches

Social constructs 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 [15,12]. [15] identifies three ways to represent roles, as *named places* in relations, as *specialisation* or *generalisation*, or as *adjunct instances.* Of these only the first and last conform to the intensional character of the counts-as relation.

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 asserted in the ABox. For some purposes this representation may be non intuitive or overly verbose. In the named places approach, roles can be represented as inverse properties. The drawback is that the relation between role player, role and context is 'flattened' to a relation between role player and role. 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 subsection 3.1).

[7] describe a methodology for defining roles in OWL as a separate ontology. Bridge axioms defined using 'canPlay' properties indicate applicability of these roles to classes in a 'base ontology'. For reasoning, the base ontology and role ontology can be integrated in a single ontology where all natural types are defined as superclasses of the roles they can play. This approach has the nice property of keeping a strict separation of roles and natural types. However, it cannot cater for the duality of role-as-class vs. role-as-relation.

The structure required for notions in social reality is very similar to one of the most prominent design patterns discussed by the Noy and Rector in a note of the Semantic Web Best Practices and Deployment (SWBP) working group: n-ary relations, relations between more than two entities [13]. One could say that a role is an n-ary relation between role player, role and context. Noy and Rector represent n-ary relations as a class with n properties that *reifies* the relation.[4] For instance, a high probability has_diagnosis relation between *Christine*, *Breast_Tumor_Christine* is represented as an individual of type Diagnosis_Relation with properties value *Breast_ Tumor_Christine* and probability *high*. 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. Association classes can only be used to reify a *binary* relation where the relation itself has additional properties (see Figure 1).

In [8] we discuss that this solution is not arbitrary: the reification makes explicit some knowledge about the domain which would be left implicit in the n-ary relation. For instance, a structured representation of an action explicates the thematic roles played by its participants. In an n-ary relation, these roles are just positions in a predicate. In other words, the restriction to binary relations in OWL enforces a more ontologically concise representation. The drawback is that exactly because of this commitment, the n-ary relation itself can no longer be represented. The question we try to answer here is whether we can devise a design pattern that can reclaim some of the relational character of the reification.

3 The Design Pattern

Current representations of social reality and n-ary relations involve an important commitment to either a relation oriented approach or a class centred representation: these options are mutually exclusive. 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. Therefore, an explicit methodology for creating (or inferring) an abstraction of

[4] [13] avoid the use of this term because of differing interpretation of the term in TopicMaps and RDF.

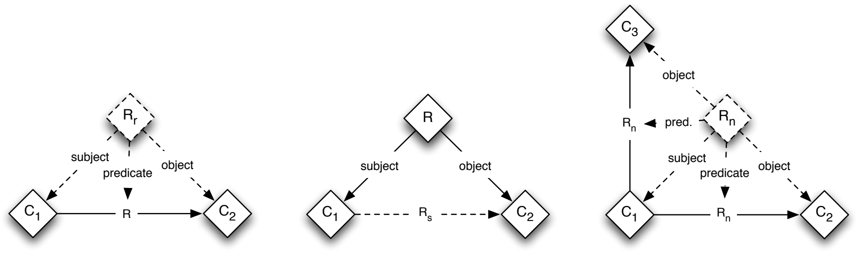

Figure 1. Reification versus abstraction R, and reification R_n of an n-ary relation.

ontologically concise, but verbose representations can be very useful for ontology reuse. A reusing ontology may adopt just the abridged version of 'properties' without compromising its ontological commitment. Figure 1 shows the reification R_r of relation R as an individual with explicit subject, predicate and object relations, and the converse abstraction R_s of individual R as a relation between classes C_1 and C_2.

The following outlines a design pattern that allows us to *infer* the abstraction of a reified relation. This pattern is based on the five design steps identified in [9,8]: create *initial class* description, *constrain* the number of role fillers, *disambiguate* role fillers, *traverse* the tree, and introduce *domain dependence*. First, we describe the general principle in terms of the construction of social roles. This is then elaborated to show its application for n-ary relations and propositional attitudes.

Consider the definition of a simple social role: a student is a person who is enrolled as such at some university. Using [14]'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 can define the Student class as follows:

$$\mathsf{Student} \equiv \mathsf{counts\_as}^{-}\ \textbf{some}\ \mathsf{Person} \sqcap \mathsf{context}\ \textbf{some}\ \mathsf{University}$$

This definition does not constrain the number of and type of entities that are valid values for the context and counts_as properties. At a more general level, a subjective entity (defined in LKIF Core) is defined as an entity imposed on another entity using a counts-as rule:[5]

$$\begin{aligned}\mathsf{Subjective\_Entity} &\equiv \mathsf{imposed\_on}\ \textbf{some}\ \mathsf{owl{:}Thing} \sqcap \mathsf{context}\ \textbf{some}\ \mathsf{owl{:}Thing}\\ &\sqsubseteq \mathsf{imposed\_on}\ \textbf{exactly}\ 1\ \mathsf{owl{:}Thing} \sqcap \mathsf{context}\ \textbf{exactly}\ 1\ \mathsf{owl{:}Thing}\\ \mathsf{imposed\_on} &\equiv \mathsf{counts\_as}^{-}\end{aligned}$$

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

[5] 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 [9]. For a more restrictive definition we could define imposed_on and context as functional properties.

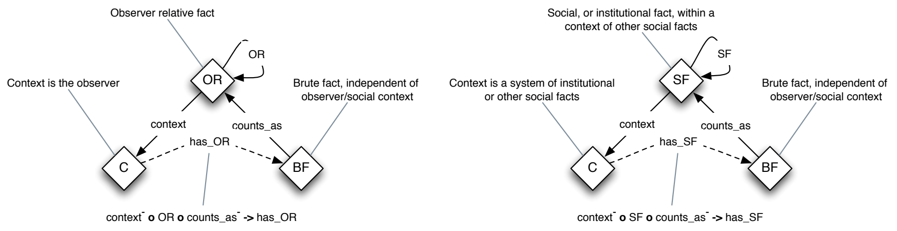
Observer relative fact
OR
OR
Context is the observer
Brute fact, independent of observer/social context
context
counts_as
has_OR
C
BF
context⁻ o OR o counts_as⁻ -> has_OR
Social, or institutional fact, within a context of other social facts
SF
SF
Context is a system of institutional or other social facts
Brute fact, independent of observer/social context
context
counts_as
has_SF
C
BF
context⁻ o SF o counts_as⁻ -> has_SF

Figure 2. Observer-relative and Institutional facts

from Subjective_Entity. The property played_by is defined as the inverse of a plays subproperty of counts_as:

$$\begin{aligned}
\mathsf{Role} &\equiv \mathsf{played\_by}\ \textbf{some}\ \mathsf{owl{:}Thing}\\
&\sqsubseteq \mathsf{Mental\_Entity}\\
\mathsf{Social\_Role} &\sqsubseteq \mathsf{Role} \sqcap \mathsf{played\_by}\ \textbf{some}\ \mathsf{Agent}\\
\mathsf{played\_by} &\equiv \mathsf{plays}^{-}\\
\mathsf{played\_by} &\sqsubseteq \mathsf{imposed\_on}\\
\mathsf{plays} &\sqsubseteq \mathsf{counts\_as}
\end{aligned}$$

If subsumed by these classes, the Student role can only be played by a single person at a single university:

$$\mathsf{Student} \sqsubseteq \mathsf{Social\_Role}$$

At this point, we have said nothing about how the classes Person and University are related.[6] 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. To prevent the University from being a student in the context of itself, the imposed_on and context properties are made disjoint.

A straightforward abstraction of this reification can be specified as role inclusion axioms that simulate the respective named places:

$$\begin{aligned}
\mathsf{context}^{-}\ \textbf{o}\ \mathsf{plays}^{-} &\sqsubseteq \mathsf{student}\\
\mathsf{plays}\ \textbf{o}\ \mathsf{context} &\sqsubseteq \mathsf{university}
\end{aligned}$$

An OWL 2 reasoner can infer a student relation between some university U and person P, and its inverse. Because in role inclusion axioms the property chain is a sub property of the property, the abstraction is subject to a similar limitation as RDF reification, but in the exact opposite direction. It can only infer the relation given the reified structure; while RDF reification only infers the reified structure, given the relation. Also, the abstraction cannot be used to identify an explicit predicate (cf. rdf:predicate) for the reified relation between the summarised class (Student) and the relation (student).

[6] The same problem as with the definition of Transaction in [9]

The role inclusion axiom will infer the relation for *any* context and brute fact connected through a Role. We therefore need to introduce domain dependence [9] and define the student relation in terms of some of the elements particular to the structure of the Student class. There are two ways to do this: either by refining the properties involved, or by using **self** restrictions. The drawback of the first 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, such properties can only be used local to the role being defined, and the properties cannot be used to infer class membership (as they need to be given in advance). A more economical approach is to introduce 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:

$$\mathsf{Student} \equiv \mathsf{is\_student}\ \textbf{some self}$$

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 abstraction axiom as follows:[7]

$$\mathsf{context}^{-}\ \mathbf{o}\ \mathsf{is\_student}\ \mathbf{o}\ \mathsf{played\_by} \sqsubseteq \mathsf{student}$$

Figure 2 shows the general structure of this pattern for observer relative and institutional facts. To complete the picture, 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 [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.

3.1 Examples

This section gives an overview of how this pattern can be applied to various other contexts: n-ary relations, propositional attitudes and speech acts. 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, see R_n in Figure 1. In fact, [13] point out, n-ary relations typically have multiple objects, but a single subject. For instance, we can extend the definition of a Diagnosis_Relation as follows:

$$\begin{aligned} \mathsf{Diagnosis\_Relation} &\equiv \mathsf{is\_diagnosis}\ \textbf{some self} \\ \mathsf{has\_diagnosis}\ \mathbf{o}\ \mathsf{is\_diagnosis}\ \mathbf{o}\ \mathsf{diagnosis\_value} &\sqsubseteq \mathsf{diagnosis} \\ \mathsf{has\_diagnosis}\ \mathbf{o}\ \mathsf{is\_diagnosis}\ \mathbf{o}\ \mathsf{diagnosis\_prob} &\sqsubseteq \mathsf{diagnosis} \end{aligned}$$

We can easily extend this pattern to express more specific kinds of diagnosis, or add informative properties such as a has_cancer_prob property that relates the

[7] The university property can be expressed by reversing the direction of the context and played_by properties.

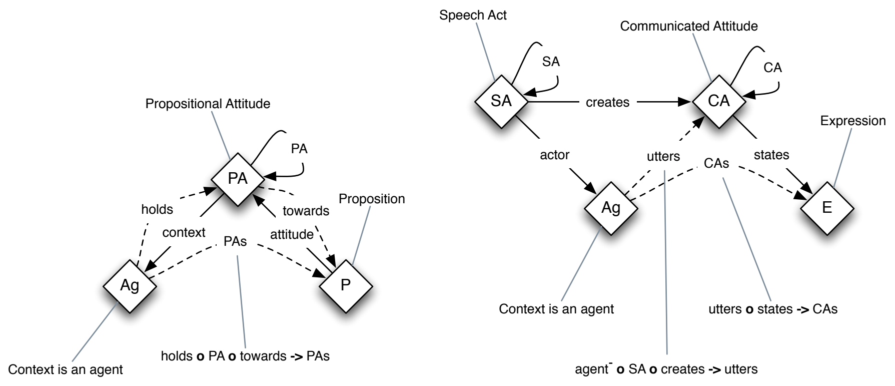

Figure 3. The propositional attitude PA held by an agent Ag towards a proposition P (left), and the communicated attitude CA that states expression E, created by speech act SA and uttered by agent Ag (right).

patient to the probability of such a particular diagnosis:

$$\begin{aligned}\mathsf{Cancer\_Diagnosis} &\equiv \mathsf{diagnosis\_value}\ \textbf{some}\ \mathsf{Cancer}\\ &\equiv \mathsf{is\_cancer\_diagnosis}\ \textbf{some self}\\ \mathsf{has\_diagnosis}\ \textbf{o}\ \mathsf{is\_cancer\_diagnosis}\ \textbf{o}\ \mathsf{diagnosis\_prob} &\sqsubseteq \mathsf{has\_cancer\_prob}\end{aligned}$$

This representation can be extended in a straightforward manner to cover more complex n-ary relations (see [8]).

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. A conviction of either murder or manslaughter hinges on the presence of an *intent* to kill. Propositional attitudes subjective entities that have a relational character very similar to roles. Consider the sentence: "Mary believes that John killed Susan.". The propositional *content* of the belief is "John killed Susan". However, 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'.".

To express these attitudes, we have to introduce the properties towards and holds as sub properties of imposed_on and context^{-}. A propositional attitude is anything held by an Agent towards some Proposition (see Figure 3). This is easily refined to a definition of Belief, its corresponding marker property, and the abstraction property chain:

$$\begin{aligned}\mathsf{Belief} &\equiv \mathsf{is\_belief}\ \textbf{some self}\\ &\sqsubseteq \mathsf{Propositional\_Attitude}\\ \mathsf{holds}\ \textbf{o}\ \mathsf{is\_belief}\ \textbf{o}\ \mathsf{towards} &\sqsubseteq \mathsf{believes}\end{aligned}$$

We successfully applied the pattern to cover more complex classes such as speech acts: actions that create a communicated attitude such as declarations, assertions and promises (see Figure 3, [8]).

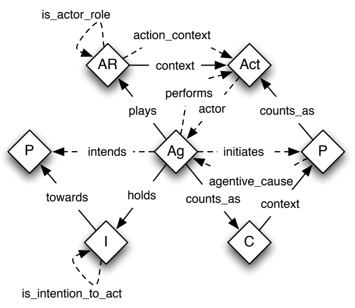

Figure 4. Four summarisation triangles used in the definition of actions [8].

4 Discussion

The preceding section gives an overview of a design pattern for representing social reality using abstractions over reified relations. This pattern relies heavily on the new features of OWL 2, namely the ability to assert property disjointness (between counts_as and context), role inclusion axioms, self restrictions and punning. It reuses modelling steps identified in earlier work [9], and we applied it to several modelling problems including the well known issue of representing n-ary relations. One application of the pattern, in [8], was the definition of the relation between actions and processes, where actions are subjective entities in the context of the agent that performs the action. This resulted in a combination of four abstraction patterns (see Figure 4). Experience with this exercise has shown that extending the combination of patterns further is hindered by the fact that the super-property of role chains is a *complex* property, which restricts the types of OWL axioms allowed (e.g. no cardinality restrictions). The problem here is that without cardinality restrictions, the pattern is no longer able to distinguish between the different fillers of the three positions of the triangle.

In the introduction to the pattern, it was mentioned that abstractions of reifications could contribute to a more practical reusability of ontologies. The broad range of use cases for the summarisation pattern discussed in this paper clearly shows that its applicability is far reaching. Without abstraction, the choice between a *relation* or *class* oriented representation of is an important ontological commitment. And more importantly, this choice would have to be made individually, for each 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 [11] and normative assessment described in [17], is quite demanding because of the large number of complex class axioms in the LKIF ontology itself (see [10,8]). Abstractions allow 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 abstraction properties, while discarding the axioms related to their reification. The resulting ontology is much more succinct, both qua ontological commitment and qua expressiveness. While abstraction 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 OWL 2 [2], but this would not be a safe reuse of the original ontology [3].

References

1. Daniel C. Dennett. *The Intentional Stance.* MIT-Press, 1987.
2. Boris Motik et al. OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax. Technical report, W3C, 2009.
3. Silvio Ghilardi, Carsten Lutz, and Frank Wolter. Did i damage my ontology? a case for conservative extensions in description logics. In P. Doherty, J. Mylopoulos, and C.A. Welty, editors, *KR*, pages 187–197. AAAI Press, 2006.
4. Davide Grossi. *Desigining Invisible Hancuffs. Formal Investigations in Institutions and Organizations for Multi-agent Systems.* SIKS dissertation series, Utrecht University, 2007. 2007-16.
5. Nicola Guarino and Christopher A. Welty. An Overview of OntoClean. In S. Staab and R. Studer, editors, *Handbook on Ontologies*, chapter 8. Springer Verlag, 2004.
6. Patrick Hayes. RDF semantics. Recommendation, W3C, 10 February 2004.
7. J. Henriksson, M. Pradel, S. Zschaler, and J Z. Pan. Ontology design and reuse with conceptual roles. In D. Calvanese and G. Lausen, editors, *Proceedings of RR 2008*, pages 104–118, 2008.
8. Rinke Hoekstra. *Ontology Representation – Design Patterns and Ontologies that Make Sense*, volume 197 of *Frontiers of Artificial Intelligence and Applications.* IOS Press, Amsterdam, June 2009.
9. Rinke Hoekstra and Joost Breuker. Polishing diamonds in OWL 2. In Aldo Gangemi and Jérôme Euzenat, editors, *Proceedings of EKAW 2008*, October 2008.
10. Rinke Hoekstra, Joost Breuker, Marcello Di Bello, and Alexander Boer. LKIF Core: Principled ontology development for the legal domain. In *Law, Ontologies and the Semantic Web.* IOS Press, 2008.
11. Szymon Klarman, Rinke Hoekstra, and Marc Bron. Versions and applicability of concept definitions in legal ontologies. In Kendall Clark and Peter F. Patel-Schneider, editors, *Proceedings of OWLED 2008 DC*, Washington, DC (metro), April 2008.
12. F. Loebe. An analysis of roles. Technical Report 6, Research Group Ontologies in Medicine, University of Leipzig, 2003. Onto-Med Report.
13. N. Noy and A. Rector. Defining n-ary relations on the semantic web. Working group note, W3C, April 2006. http://www.w3.org/TR/swbp-n-aryRelations/.
14. John R. Searle. *The Construction of Social Reality.* New York, 1995.
15. Friedrich Steimann. On the representation of roles in object-oriented and conceptual modelling. *Data and Knowledge Engineering*, 35(1):83–106, October 2000.
16. A. Valente. *Legal Knowledge Engineering: A Modelling Approach.* PhD thesis, University of Amsterdam, Amsterdam, 1995.
17. S. van de Ven, R. Hoekstra, J. Breuker, L. Wortel, and A. El-Ali. Judging Amy: Automated legal assessment using OWL 2. In *Proceedings of OWLED 2008 EU*, October 2008.