ESTRELLA	ESTRELLA IST-2004-027655 ***European project for Standardized Transparent Representations in order to Extend LegaL Accessibility Specific Targeted Research or Innovation Project*** Specific Targeted Research Project Information Society Technologies

Deliverable N°: 4.6

Developing HARNESS Towards a hybrid architecture for LKIF

Version:	FINAL 1.0
Due date of Deliverable:	September 30, 2008
Actual submission date:	September 30, 2008
Start date of Project:	1 January 2006
Duration:	30 months
Project Coordinator:	Universiteit van Amsterdam (NL)
Lead contractor deliverable:	University of Amsterdam
Participating contractors:	Alma Mater Studiorum - Universita di Bologna (IT), University of Liverpool (UK), Fraunhofer Gesellschaft zur foerderung der angewandten forschung e.v. (DE), RuleWise b.v. (NL), RuleBurst (EUROPE) Ltd. (UK), knowledgeTools International GmbH (DE), Interaction Design Ltd. (UK), SOGEI - Societa Generale d'Informatica S.P.A. (IT), Ministro per le Riforme e le Innivazioni nella Publica Amministrazione (IT), Hungarian Tax and Financial Control Administration (HU), Budapesti Corvinus Egyetem (HU), Ministero dell'Economia e delle Finanze (IT), Consorzio Pisa Ricerche SCARL (IT)

Project funded by the European Community under the 6th Framework Programme

	Dissemination Level	
PU	Public	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	

Executive Summary

This Deliverable is a complementary one, i.e. it is not specified as a Deliverable in the TA. It is the result of insights gained since the last review on the issue of hybrid architecture and work on LKIF. The Deliverable consists of two parts:

Software A ‘plug-in’ and knowledge bases that are used for testing an approach that is a prototype for experimenting with normative legal reasoning using OWL 2 DL,

This report containing the motivation and technical description of the achievements.

The research reported is largely experimental and explorative. The original goal – an architecture that could handle ontologies expressed in OWL-DL and rules in LKIF-rules – appeared not feasible given the time and effort available. In fact, such a solution is not possible if one wants to have decidable (sound and complete) reasoning. We were caught in the classical trade-off between expressivity and decidability in knowledge representation (KR) [Levesque and Brachman, 1985]. Only recently – in the development of a suitable complementary rule formalism for OWL – it has become apparent that this trade-off also applies to combinations of KR formalisms, even if the formalisms are used for different types of knowledge in different stages of the reasoning process. The upshot is simply this: by starting with an ontology cast in OWL-DL, the rule formalism has to respect the semantic constraints under which the OWL reasoner has delivered its reasoning results: the rules have to be ‘DL-safe’ (see [Motik et al., 2005], and also `http://www.w3.org/2005/rules/wiki/SWC`). This would result in a hybrid architecture where only a small subset of LKIF rules could be used. These insights are rather recent and focussed around the work on Semantic Web standards (‘recommendations’) about a next version OWL (OWL 2) [1] and about rule formalisms (see`http://www.w3.org/2005/rules/wiki/RIF_Working_Group`). While we investigated these issues, another approach was proposed due to experiences we had in using the LKIF-Core ontology to model norms in the European Directive on economic transactions, as reported in Deliverable 2.5 [Breuker et al., 2008b]. LKIF-Core is cast in OWL-DL and it was a relatively small step to to use the definitions of norms to model norms in OWL: a first description can be found in the Appendix of this report. Hybrid reasoning, i.e. combining a knowledge base that contained norms and a knowledge base that contained terminological knowledge (ontology) became probably attainable. This approach was further investigated. The advantages became

[1]The Leibniz Canter for Law of the UvA participates in the OWL 2 W3C committee.

immediately apparent: we could use the same DL reasoner (Pellet) to handle both the ontology and the legal norms for assessing legal cases. Moreover, this reasoning could be performed simultaneously with both knowledge bases, and this reasoning had all these nice properties as to be proven complete and decidable. Experiments were performed and the initial results were very encouraging. Moreover, it looked that such a solution are innovative in two respects:

1. Thus far, OWL reasoning has been almost exclusively focussed on the use of ontologies (terminological knowledge) and we could show that by fully exploiting OWL 2's expressiveness also normative 'rules' could be handled in combination with a domain ontology. This result was reported at the OWLED-2008 workshop which is devoted to research of OWL applications [van de Ven et al., 2008b].

2. It has always been assumed that reasoning with norms required non-monotonic inference (cf. deontic logic; research in AI & Law). We demonstrate that this is not necessarily the case [van de Ven et al., 2008a].

Five different approaches were taken. We report here the two most promising ones in Chapter 3. Experiments were performed with two legal domains: 1) a toy example – a university library regulation – with complex structures as a real test case and 2) part of the Hungarian code about taxation of gifts, as a more realistic domain.

What we report are still results from ongoing research. While we can show that we can perform basic deontic reasoning in OWL-DL there is still a dangling problem that in some circumstances we will not be able to infer the identity of an individual: more specifically: we cannot force OWL to *derive* from a generic description ('T-Box') that two individuals ('A-Box') are the same one. [2] This problem is a general one, i.e. not typical for legal domains. We have developed a 'design pattern' that provides an important 'approximate' solution [Hoekstra and Breuker, 2008, Hoekstra, 2008], but we do not know yet whether this problem occurs frequently or systematically in legal case descriptions, and whether the approximate solution holds in these cases. In other words: we need more research on these issues. This research has recently started as part of a Dutch research project (AGILE) of the Leinbiz Center for Law (University of Amsterdam), the Technical University Delft and two vendors of legal support systems as end-users.

The solution we present here is not a real architecture: it is only a component of a future architecture. The aim is still to combine this solution with reasoning with rule formats, as we can foresee that, even if all deontic reasoning can be handled by a DL reasoner, we will also need rules for more complex legal reasoning tasks. Legal normative assessment as implemented in this 'proto-HARNESS' solution is a kernel subtask in all legal reasoning tasks, such as legal planning, drafting regulations and legal argumentation (dispute). The advantage of this OWL-DL solution for normative reasoning over traditional rule based solutions is that we 1) can exploit well tested and state of the art Semantic Web technology, which is freely available, and 2) this technology enables automated reasoning that can be trusted blindly.

[2] We can assert so, but that is not what is at stake.

Deliverable 4.6, Developing HARNESS,
Towards a hybrid architecture for LKIF

Joost Breuker and Saskia van de Ven (editors),
Abdallah El-Ali, Marc Bron, Rinke Hoekstra, Szymon Klarman and Uroš Milošević and Lars Wortel
Leibniz Center for Law
Faculty of Law
University of Amsterdam
The Netherlands

Andŕas Förhécz
Corvinus University of Budapest
Department of Computer Science
Hungary

In this Deliverable we report the development of the HARNESS architecture [a]. As will become apparent from Chapter 2, due to fundamental problems in aligning a DL (Description Logic) reasoner wit a rule engine, we arrived halfway with a solution that combines reasoning with an ontology and a normative knowledge base, both expressed in OWL 2 DL. This solution which is described in detail in Chapter 3 is an innovative and unexpected one. It does not really deserve the label 'architecture' as it does not consist of various components. This report is complementary to the software delivered, i.e. a plugin for Protégé-4 and knowledge bases used for testing the approach described in this document. HARNESS is aimed at the legal assessment of cases.

[a]For: Hybrid Architecture for Reasoning with Norms Exploiting Semantic web Standards.

Contents

ESTRELLA
p/a Faculty of Law
University of Amsterdam
PO Box 1030
1000BA Amsterdam
The Netherlands

tel: +31 20 525 3485/3490
fax: +31 20 525 3495
http://www.estrellaproject.org

Corresponding author:
Joost Breuker
tel: +31 20 5253494
breuker@science.uva.nl
http://www.leibnizcenter.org/
information/people/joost-breuker

Contents

Chapter 1

Introduction

1.1 Legal assessment: a core task in legal reasoning

In this Deliverable we report the development of the HARNESS architecture. [1] The development of a hybrid architecture for LKIF was proposed and recommended at the last Estrella review. As will become apparent from Chapter 2 due to fundamental problems in aligning a DL (Description Logic) reasoner with a rule engine, we arrived halfway with a solution that combines reasoning with an ontology and a normative knowledge base, both expressed in OWL 2 DL `athttp://www.w3.org/2007/OWL/`. This solution is an innovative and unexpected one. At first sight, OWL 2 DL appears not very suitable for reasoning with norms. DL reasoners are monotonic and deductive, and it is generally assumed that (legal) norms require some form of non-monotonicity: either built in in the representation formalism (as in LKIF-Rules) or by meta-level control. The solution, which will be described in detail in Chapter 3, is still an experimental one. It does not really deserve the label 'architecture' as it does not consist of various components. It is hybrid but covers only one knowledge representation formalism: OWL 2 DL. However when we refer in this report to HARNESS we usually mean this to be the OWL 2 DL based component of this prospective architecture.

This report is complementary to the software delivered, i.e. a plugin for Protégé-4 and knowledge bases used for testing the approach described in this document. HARNESS is aimed at the legal assessment of cases. It is based on the notion that (legal) norms express generic situations in which something (state, action) is undesirable (prohibited) or prescribed. A case is a description of an actual situation. If a norm matches some part of the description of a legal case, then a norm is applicable, which may result in the assessment whether that (part of the case) is compliant with, or violates a norm. Legal assessment is the core of practical legal reasoning, i.e. all typical legal reasoning tasks, such as designing norms (legislation, contracts) or planning legal actions or constructing legal arguments have legal assessment as a subtask. Legal assessment checks legal consequences which is the test to find out whether drafted norms or planned actions are legally as intended. Similarly, legal arguments about cases cannot be valid when they are not in accordance with valid, applicable norms.

[1]For: Hybrid Architecture for Reasoning with Norms Exploiting Semantic web Standards.

1.2 Motivation for developing HARNESS

LKIF – the Legal Knowledge Interchange Format developed in Estrella – has two main roles: 1) to provide an interchange format for knowledge representation (KR) formalisms for legal knowledge systems and 2) to enable developers to build legal knowledge systems using LKIF [Klarman, 2008], [Gordon, 2006]. The first role has been investigated, realized and reported in Estrella (WP-2) [Breuker et al., 2008b]; for the second role Carneades has been used to run LKIF-Rules and LKIF-Argument. However, to enable also the use of legal ontologies as a knowledge base cast in OWL-DL two kinds of hybrid solutions are available:

1. To translate OWL-DL into rules: this is reported in Deliverable 4.3. This translation comes however with a price. To maintain the semantic restrictions specified in OWL-DL into the rule environment, only a specific subset of DL expressivity can be used: rules that comply with a DLP (Description Logic Programming) subset [Grosof et al., 2003]). Note that this restriction not only applies to the 'ontology' rules but also to all rules, if one wants to maintain the soundness, completeness and decidability implied by the DL semantics.

2. To construct a hybrid architecture using different formalisms in which different subtasks are handled by different reasoners. In our case, it means that a legal case description, represented as individuals and their properties, is expanded by implications from the corresponding ontology via a DL reasoner. This expanded – i.e. semantically interpreted case description is then fed into a rule engine where norms in the form of rules are applied to yield normative qualifications. However, this construction meets similar (but not the same!) problems as the translation solution as we will explain in Chapter **??**.

We focus here on the knowledge representation (KR) formalisms. There is some correspondence between the *types of knowledge* and KR formalisms, but they should not be confounded. For instance, DL based formalisms are very suitable for expressing terminological knowledge: ontologies. This is also suggested by the O in OWL, which stands for ontology. OWL is capable of representing and reasoning with other kinds of knowledge as well, as we will show in this report. DL based KR has a long history in Artificial Intelligence (AI) as a generic KR approach. It is also possible to represent terminological knowledge (ontologies) by means of a rule formalism. However, it appears that representing concepts (terms) as classes is not only more 'natural' in modeling, but for the DL based representations sound, complete and efficient reasoners, which are not (yet) available for rule based approaches. On the other hand, rules make modeling of – and reasoning with – other types of knowledge than terminological knowledge much easier, due to the fact that they have variables, and OWL has not. Therefore, combining rule and class based KR seems to be a very attractive, 'hybrid' solution in representing and reasoning with different types of knowledge. Distinguishing types of knowledge and distributing the representation and reasoning over various models has been the Leitmotiv of knowledge engineering methodology for over twenty years ([Schreiber et al., 1993, Schreiber et al., 2000]; for an overview of types of knowledge and their dependencies in legal reasoning see

[Valente, 1995a]). Ontologies – terminological knowledge – play a very prominent role in semantic (web) based information management. This role is in fact by far the dominant one. It is easily overlooked that ontologies are in the first place 1) knowledge bases, which 2) have a primary role in goal directed reasoning (problem solving). This role is to enable a problem solver to 'understand' a problem situation, as we will explain in the next section.

Ontologies as the backbone in knowledge based reasoning Starting with OWL is motivated by the fact that ontologies have a special status in reasoning. Ontologies represent that kind of knowledge that is considered to be the undisputed backbone of what is known about a particular domain. It represents the terminology – i.e. an axiomatic starting point on which domain knowledge is built – and it is assumed that it is the necessary foundation for the various kinds of knowledge that may appear to be useful in the domain. The distinction between knowledge and belief in a domain is often fuzzy. However, this terminological knowledge – ontology – is assumed to be true, and specifying this grounding by the use of definitions is viewed as *ontological commitment*, i.e. setting this knowledge apart to provide a knowledge resource for the feeding of all kinds of reasoning. It is 'shared' by other kinds of knowledge and belief. Note that this is based upon an assumption. The validity of this assumption is not only shown by the fact that it indeed is shared by these other kinds of knowledge, but also by the fact that persons that are knowledgeable about the domain agree that they hold this knowledge to be true. This agreement is not a question of voting or explicit statement. In fact, the agreement is silent, as it is undisputed and applied without justifications in human reasoning methods and natural language understanding. One is usually not aware of the use of this knowledge, although it may be the major initial component in acquiring knowledge of a domain. Educational introductions contain largely explanations of the major concepts, reflecting the primacy of this kind of knowledge. Given this (widely shared) view on the role of ontologies, one would expect that knowledge systems would contain ontologies as their basic resource for reasoning. That is unfortunately not the case; not only in the the legal domain. [2]

This view on the role of ontologies – as a terminological knowledge knowledge base – has been the principal 'use case' in the development of OWL DL. In OWL, expressiveness is sacrificed for decidability to enable sound and complete reasoning (see [Horrocks et al., 2003] for a detailed account of OWL's design decisions). The sublanguage OWL DL is so named due to its correspondence with Description Logics, a field of research that has studied a particular decidable fragment of first order logic. Research in DL has lead to relatively highly expressive formalisms maintaining decidability. OWL-DL is such a result, but as the work on OWL 2 DL shows, still some gains in expressiveness are attainable. [3]

[2]Of course there are exceptions. For instance, in Qualitative Reasoning [Forbus, 2008] and model based reasoning ontologies have played this role already for a few decades.

[3]The Leibniz Center for Law is involved in this development. For the current state see: www.w3.org/2007/OWL/wiki/Primer. Note that OWL 2 tools are already available (e.g. Protege; TopBraid, Pellet, KAON,...). In this report the default meaning of OWL is the OWL-DL 2 species.

Chapter 2

A hybrid approach

There are two views on what 'hybrid' means. In the early days of DL based KR (early 90-ies) a hybrid reasoner combined rules with DL reasoning. For instance, the CLASSIC system combined DL reasoning with special kinds of rules called procedural attachment [Brachman et al., 1991]. These rules were added to calculate values etc. However current research aimed at combining DL reasoners with rules define a somewhat different and more precise combination of rules and ontologies. In a survey on combining rules with ontologies [Antoniou et al., 2005, p 3]:

> "In the hybrid approach there is a strict separation between the ordinary predicates, which are basic rule predicates and ontology predicates, which are only used as constraints in rule antecedents. Reasoning is done by interfacing an existing rule reasoner with an existing ontology reasoner."

Note that it is here ontology combined with rules, which means that it is not necessarily the case that two different formalisms are used: in this sense Carneades as described in Deliverable 4.3 is a hybrid reasoner. However, current research in the context of Semantic Web technology takes another approach and rather starts from an ontology expressed in a DL formalism and searches for compatible rule formalisms. In fact, combinations have been found that can be so intimate with OWL-DL that the same reasoner can handle the rules and the ontology at the same time, provided the rules are specified as DLP (Descriptive Logic Programming) rules. DLP is a fragment of DL that is compatible with a decidable fragment of Disjunctive Datalog. In this way the highly efficient tableau algorithms of (most) DL reasoners can be fully exploited. In fact, DLP is now even defined in the OWL2 proposal as a special 'profile': OWL 2 RL. [1]. However, this is not the way we have pursued our search for hybrid solutions, as the DLP rules are not very expressive (e.g. they do not allow negation). DLP rules cover only a subset of DL (see Figure 2.2). Therefore we wanted to maintain as much as possible the expressiveness of OWL-DL (and in particular: that of OWL 2 DL), and combine this with rules in a way in which we could optimize expressivity and still preserve soundness and completeness.

In the definition of 'hybrid' quoted above, it is the rules that do the work and the ontology provides semantic *constraints*. For instance, if a rule expresses the norm that vehicles should keep to the right of the road, the ontology may define 'right-of' as 'not(left-of)' and thus being able to conclude that a car (= a vehicle)

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

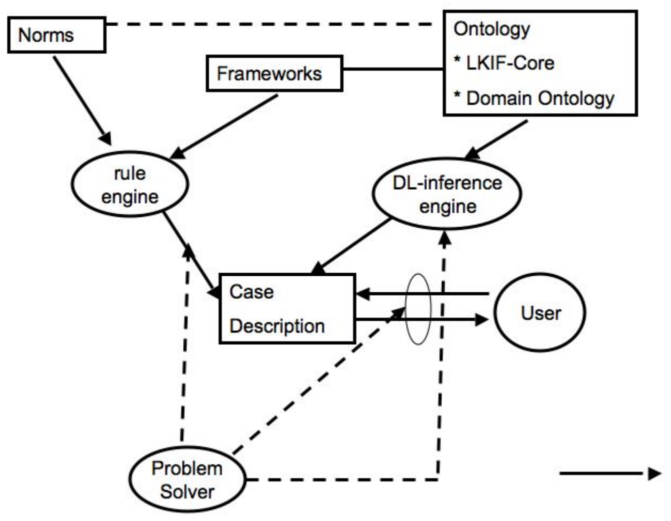


Figure 2.1: Main components of conceptual architecture

violates this rule riding on the left hand side of the road. Note that these constraints can either be viewed as applicable to the description of a particular situation, i.e. provide a semantic interpretation to this situation, or as as an extension to the rule antecedents: the rule antecedent will not only contain the term 'right-of' but contains now ((right-of) or (not(left-of))). Discovering these constraints is essentially what a classifier like Pellet does. This is similar to the way in which ontologies and 'model fragments' (rule patterns) work together in Qualitative Reasoning [Forbus, 2008].

The initial hybrid architecture proposed is summarized by Figure 2.1. The hybrid reasoning should be performed in two steps:

1. A case description by the user in terms of individuals and their classes plus properties describing the case would be classified by Pellet, and all inferences should be added; the Case Description is an A-Box in DL terms. For instance, if it was specified that car1 was at left of car2 in describing a traffic situation, it would also be derived that car2 was at the right of car1. This is important for example, to be able to match norms that specify that vehicles coming from the right have priority.

2. This extended case description is submitted to the rule engine that applies norms. We leave out further details of the normative reasoning involved, as they will re-appear in Chapter 2.2.

2.1 Theoretical and practical problems in a hybrid reasoning

The architecture proposed implies that the content of the A-Box has to be translated in a format digestible for the rule engine. A translator from OWL to the s-expression syntax which for instance is used by Carneades was already available (see

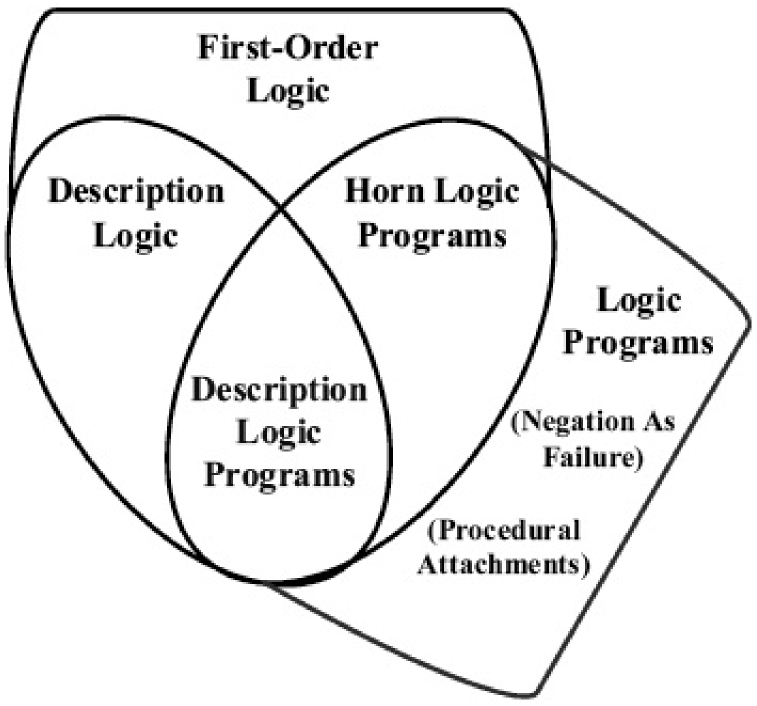


Figure 2.2: Venn diagram representing KR formalisms wrt FOL expressivity

[Breuker et al., 2008b]) and could be used for this purpose. However, at that time a practical problem was that Pellet's (default) output did not contain all its inferences. The most serious problem for a hybrid approach we have already mentioned: it is the fact that the semantic constraints provided by Pellet should be respected by the rule formalism, as otherwise the guarantee for sound and complete reasoning may get lost. To prevent tampering with the DL semantics the rules that are complementary to OWL-DL should be 'DL-safe', meaning that they should be also constrained in expressiveness similar to OWL-DL [Motik et al., 2005, Rosati, 2006]. [2] Figure 2.2 provides in a nutshell the position of various rule formalisms with respect to DL and FOL. [3]

This constraint on semantically compatible rule formalisms begs the question why one would want to combine these two knowledge representation (KR) formalisms

[2] "DL-safety only restricts the interchange of consequences between the component languages to those consequences involving individuals explicitly introduced in the ABox. Note that DL-safe rules are still more expressive than DLP rules." [Motik et al., 2005]. This means al least that non-montonic operators (e.g. 'unless') and negation are not allowed; plus that no new individuals are introduced for which there are no classes, etc. Moreover, the rules should be restricted to known individuals to safeguard decidability. In other words: DL-safe is rather restrictive. The precise definition of DL-safe is still subject of research: see `http://www.w3.org/2005/rules/wiki/SWC`

[3] FOL is undecidable. There is a strong correlation between decidability and tractability: tractability is not a logical, formal property, but a computational one as studied in complexity theory. Intractablity says that scaling up may lead to unattainable computing resources; decidability refers to the fact that a formalism misses out necessary inferences (proof).

anyway. In Estrella the motivation is that we want to cover as much as possible the full range of LKIF's KR formalisms in one rational architecture that allows ontologies cast in OWL to be used in rule based legal reasoning. The requirement that the rule formalism should be DL-safe already limits the constructs in LKIF rules drastically.

However, an important reason to search for a suitable DL compliant solution is that rules enable constructs that are difficult or even impossible to model in OWL 2 DL. The most important one for our purposes is that is that OWL does not know about variables. There is some mapping between classes and individuals on one hand, and variables and constants in rules on the other hand, but variables may take varying values (variable binding) while classes can only refer to sets of individuals. This makes it difficult to have a traceable representation of changes in situations and more in general maintaining or enforcing identity of individuals.(for more details see [Hoekstra and Breuker, 2008]). This is particularly relevant if we want to maintain the identity of individuals under change. Searching for an appropriate rule companion for OWL is also a long time concern for the W3C committee on rules for the Semantic Web. An initial candidate, SWRL has been abandoned and after about four years of work, the current proposal is an interchange format rather than a specific formalism: RIF (see`http://www.w3.org/2005/rules/wiki/RIF_Working_Group`). This interchange format has a similar role as LKIF: to enable translation between rule formalisms. However, RIF should also be able to be combined with OWL and the DL-safeness condition - this is tentatively defined, but explicitly left further to implementation experiences (see `http://www.w3.org/2005/rules/wiki/SWC` and `http://www.w3.org/TR/rif-core/`).

Another reason to have also a rule based formalism as part of a reasoning architecture is a more pragmatic one. It appears to be easier to express our knowledge in the form of rules. Indeed modeling knowledge in OWL-DL is not as easy as in rules (as is also confirmed by the modeling experiences reported in [Breuker et al., 2008b]). This advantage is not without amendements. It is certainly true for small rule bases but not necessarily for large sets of rules, where one may easily get lost, as they lack in principle semantic indexing: the OWL graphical rendering easily conveys the structure of a knowledge base. Moreover the effort may be payed back by the fact that consistency can be checked.

2.2 Exploiting OWL for reasoning with norms

While considering the problems posed by a hybrid solution, it dawned that legal norms could also be represented as classes in OWL. OWL is in principle as much a 'universal' KR language as some rule formalism. We had already experience in expressing norms in OWL. It is described in [Breuker et al., 2008b] how norms to be expressed in LKIF- rules were modeled in OWL using the LKIF-Core ontology that contained the definitions of LKIF rules. These rules, i.e. the norms of the European Directive on economic operations, were modeled in terms of the ontology of this Directive, and were translated (in a rather complex way) to rules cast in s-expressions that could be digested by Carneades. These norms were represented as individuals of the LKIF-rule classes in the LKIF-Core ontology. This observation inspired us to use OWL to create complex classes that represented norms in such a way that:

1. The classifier could classify descriptions of individuals (a case description) as belonging to a norm class, i.e. the equivalent of matching an antecedent of a rule to a description in terms of constants. If the classification succeeds, a deontic qualification of the norm can be attributed to (a part) of the case description, and it can be derived whether this part belongs to the class of 'allowed' or 'disallowed'

2. Moreover, the classifier can classify the norms in a subsumption hierarchy which is the basis for discovering which norms are exceptions to which other norms. In other words we get almost for free the solution to a serious problem in modeling norms: the exceptions are discovered by the reasoner, where in rule based approaches they have to be identified by the modeler (e.g. using the LKIF-rule construct 'unless') or by debugging inconsistencies. [4])

The detailed proposal to include reasoning with norms cast in OWL-DL is added to this report as an Appendix. This looks as an 'all OWL' solution, but the finishing touch of the reasoning is provided by the use of SPARQL, a query language for OWL and RDF. Its role here is to enable rule-like assertions. It is used to select the most specific norm when more than one norm classifies the same part of a case description. In fact, more in general, one can use DL non-safe rules as long as the effects do not modify anything stated (constrained) by the OWL reasoner. Since then, the focus has been on the exploitation of OWL to combine reasoning with the ontology and the norms. The most worked out explorations are reported in the next Chapter. It should be noted that by reporting these exploitations we do not think that the foreseen hybrid architecture of HARNESS is abandoned. It is not for several reasons:

- The experiments, as reported in Chapter 3, make clear that some 'finishing touch' is still required. Pellet is a monotonic reasoner that cannot handle the inconsistencies inherent in 'exceptions' to norms. By representing norms in such a way that these inconsistencies do not appear as such, these hidden inconsistencies in the output of Pellet can be easily solved by an external selection of the preferred, i.e. most specific norm applicable. This 'meta-level' selection can be written in a few simple rules.

- It is (still) not sure whether the identity-of-individuals problems – they appear in different versions – can sufficiently be 'approximated' by the solutions proposed [Hoekstra and Breuker, 2008]. If not, we need also variables, i.e. rules.

- OWL is part of LKIF in the first place because legal ontologies are usually written in OWL. In practice these ontologies are used for information retrieval purposes, while the legal knowledge systems developed use some rule formalism. A hybrid architecture would enable these vendors to convert and combine these knowledge bases into more powerful and reliable knowledge system.

[4]This is is a short cut description of problems involved in inconsistencies in norms. For more details see [Boer, 2008]

In the next Chapter we present the main results of the experiments performed, all based on the same starting point: the representation of norms as classes, as originally specified as a first working document in developing HARNESS (see Appendix 4).

Chapter 3

Exploring the power and limitations of OWL

In this Chapter we will present our approach in more technical detail. We will elaborate on how we intend to use OWL 2 DL to express norms and how normative qualification takes place, illustrated by examples from the 'taxation on gifts' domain. Three knowledge bases are needed for this task. A domain ontology defines the concepts used as the basic terminology for case descriptions and norms. The norms represent the regulations. The individual case description represents the facts that describe a situation and is expressed as a set of related OWL individuals.

In Section 3.1 we will start with explaining the usecase from the 'taxation on gifts' domain and elaborate on how we describe norms as classes in OWL. Moreover we will show how the matching between the ontology, the norms and the individual use case takes place. We will explain how we deal with exceptions between norms in a monotonic way. Moreover an outcome in terms of the normative qualification of the case is desired. This outcome needs to be explained e.g. why does a certain norm applies to the given input, showing which terms in a norm match the individual facts of the case. We have developed a Protégé 4 plugin, that assists users in this task and the general task of legal assessment. This plugin can be seen as a HARNESS prototype and is described in Section 3.4.

Section 3.2 describes a possible drawback of the "only-OWL" approach, which we refer to as the identity problem of individuals. In Section 3.3 an alternative solution to describing norms is presented.

3.1 Classification and norm application

We will illustrate our approach by means of a real-world example of the Hungarian regulations dealing with duties on gifts. In modeling this domain we focused a part of section 11, which states the following:

(1) The following shall be subject to duty payment on gifts:

   a) gifts of real property,

   b) gifts of movable property,

   c) gratuitous creation of a right of pecuniary value, surrender of such right or the exercise thereof without consideration, and the waiver of such right without consideration.

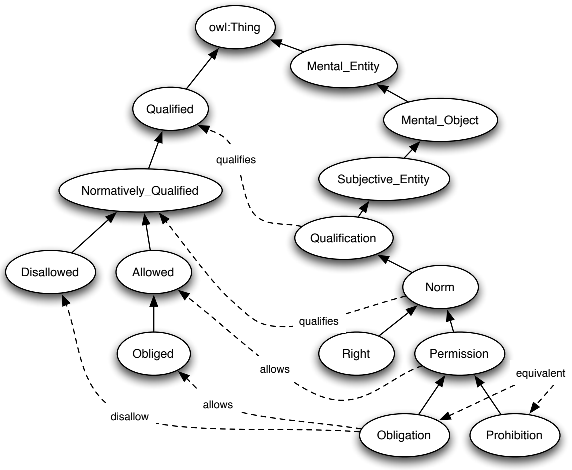


Figure 3.1: Qualifications and Norms

(2) The gifts listed under Subsection (1) shall be subject to the payment of duty only if duly documented, or, in respect of movable property, if the market value of the movable property granted to any one donee exceeds 150 000 HUF even not documented. Such gifts shall be reported to the state tax authority within 30 days in accordance with the provisions set forth in Subsections (3)-(4) of Section 91.

(3) Rights of pecuniary value given as a gift shall not be subject to the payment of duty if retained by the donor for his own benefit, or if the real property is gifted as encumbered with some rights of pecuniary value already existing, and registered in the real estate register, prior to the transfer.

Our use case deals with a donation of a copyright. Phil donates the copyright on his book to Mary. This donation is documented, but is retained by Phil for his own benefit:

$$\begin{aligned}&\mathsf{Donation}(\mathit{donation\_of\_copyright}) \ \wedge \mathsf{Documented}(\mathit{donation\_of\_copyright}) \ \wedge\\ &\mathsf{donee}(\mathit{donation\_of\_copyright}, \mathit{mary}) \wedge \ \mathsf{Person}(\mathit{mary}) \wedge\\ &\mathsf{donor}(\mathit{donation\_of\_copyright}, \mathit{phil}) \wedge \ \mathsf{Person}(\mathit{phil}) \wedge\\ &\mathsf{retained\_by}(\mathit{donation\_of\_copyright}, \mathit{phil})\end{aligned}$$

We represent a norm as a deontic qualification of a generic case [Valente, 1995b] (GC), which is a conjunction of conditions that together form a description of the situation expressed by the norm. A GC Σ is defined as a set of conditions $\{\sigma_1, ..., \sigma_n\}$ in conjunctive normal form. Conditions are class axioms composed of classes and properties defined by the domain ontology. An individual case C is a set of grounded propositions $\{c_1, ..., c_n\}$, called circumstances, that describe a certain state of affairs.

The norm itself is normatively qualified using the three deontic notions Permission, Obligation and Prohibition. The LKIF Core ontology defines the class Norm and its subclasses Permission, Obligation and Prohibition as follows [Hoekstra et al., 2007,

Notion	Match	Default Allowed	Default Disallowed
Prohibition	**no**	no effect	***possible effect***
	yes	has effect	no effect
Permission	**no**	no effect	has effect
	yes	no effect	has effect
Obligation	**no**	***possible effect***	no effect
	yes	has effect	has effect

Table 3.1: Behaviour of deontic notions with respect to the default

Boer et al., 2005]:

$$\begin{aligned}
\mathsf{Norm} &\sqsubseteq \mathsf{Qualification} \sqcap \mathsf{qualifies}\ \textbf{some}\ \mathsf{Normatively\_Qualified}\\
\mathsf{Permission} &\sqsubseteq \mathsf{Norm}\\
&\equiv \mathsf{allows}\ \textbf{some}\ \mathsf{Allowed} \sqcap \mathsf{allows}\ \textbf{only}\ \mathsf{Allowed}\\
\mathsf{Obligation} &\sqsubseteq \mathsf{Permission}\\
&\equiv \mathsf{allows}\ \textbf{some}\ \mathsf{Obliged} \sqcap \mathsf{disallows}\ \textbf{some}\ \mathsf{Disallowed}\\
&\quad \sqcap \mathsf{allows}\ \textbf{only}\ \mathsf{Allowed} \sqcap \mathsf{disallows}\ \textbf{only}\ \mathsf{Disallowed}\\
\mathsf{Prohibition} &\equiv \mathsf{Obligation}
\end{aligned}$$

The Prohibition and Obligation are equivalent because they are simply two different ways to put the same thing into words: a prohibition to smoke is an obligation not to smoke. The Permission is different in that it allows something, but does not prohibit anything. For the purpose of clarity we will refer to the subclasses of Normatively_Qualified as generic cases. The properties allows and disallows are disjoint sub properties of qualifies; a norm cannot simultaneously allow and disallow the same situation. Although a GC describes an entire situation, it can be represented as a single class description because the entities in a single situation should be connected. On the other hand, this means the norm's qualification is biased to the class that forms the entry point for describing the GC.

As the goal of our system is normative assessment, we would like to detect violations of the *default situation*. The usual normative default in law is a (weak) permission, i.e. allowed in our terms [1]. Table 3.1) provides an overview of the behaviour of the deontic notions with respect to the default situation. Most matches have simple and direct outcomes: for example matching permissions always yield 'allowed'. However there are two more complicated situations to deal with, which are listed in the table as "possible effect". If there is no match with the GC of an obligation where the default is allowed, we might have failed to detect a violation of the default. In this case, the GC is partitioned into that part which is explicitly *allowed* (A_Σ) and that part which is explicitly *disallowed* (D_Σ) by the obligation. Moreover a norm is typically only addressed towards a certain element of the GC, given some *context*. We therefore distinguish between the context of a norm and

[1] However, some regulations take as their default that everything is prohibited as a starting point. This is e.g. the case where the regulation gets the character of an instruction: for instance for emergency procedures, but the classical toy domain of a library regulation has the same character.

that part of the GC which the norm is directed towards. The *design pattern* we follow in the representation is therefore as follows:

$$\begin{tikzcd}
 & A_{\Sigma} & \equiv \sigma_1 \sqcap ... \sqcap \sigma_n \sqcap \sigma_{n+1} \\
 & & \equiv \Sigma_1 \sqcap \Sigma_2 \\
O_{\Sigma} \arrow[ruu] \arrow[rdd] & & \\
 & & \\
 & D_{\Sigma} & \equiv \sigma_1 \sqcap ... \sqcap \sigma_n \\
 & & \equiv \Sigma_1
\end{tikzcd}$$

where the context is defined as Σ_1: $\sigma_1 \sqcap ... \sqcap \sigma_n$, and the obliged part of the GC is Σ_2: σ_{n+1}. When an individual case does not match this specific (obliged) part, but it *does* match its context, an obligation is violated which leaves us with a disallowed situation. What is allowed is being compliant to the obligation: a match with the context as well as with the obliged part.

The other more complicated option in the table is the one where the default situation is disallowed and no match with a prohibition is found. We then get the following design pattern:

$$\begin{tikzcd}
 & D_{\Sigma}(\mathrm{C}) & \equiv \sigma_1 \sqcap ... \sqcap \sigma_n \sqcap \sigma_{n+1} \\
 & & \equiv \Sigma_1 \sqcap \Sigma_2 \\
F_{\Sigma}(\mathrm{C}) \arrow[ruu] \arrow[rdd] & & \\
 & & \\
 & A_{\Sigma}(\mathrm{C}) & \equiv \sigma_1 \sqcap ... \sqcap \sigma_n \\
 & & \equiv \Sigma_1
\end{tikzcd}$$

where the context is defined as Σ_1: $\sigma_1 \sqcap ... \sqcap \sigma_n$ and the prohibited part of the GC is Σ_2: σ_{n+1}. When an individual case does match with both the specific (prohibited) part, as well as its context, a prohibition is detected which leaves us with a disallowed situation, similar to the default in this case. What is allowed and thus violates the default, is a match with the context, but not with the prohibited part.

We will now take a look at how the first norm relevant for the use case is modeled. Section 11-1c, combined with Section 11-2 is modeled as follows:

$$
\begin{array}{rcl}
\mathsf{GC\_11\_1c\_P} & \sqsubseteq & \mathsf{Generic\_Case} \sqcap \mathsf{allowed\_by}\ \textbf{value}\ \textit{section11\_1c} \\
 & \equiv & \mathsf{gratuitous\_creation\_surrender\_or\_waiver\_of\_a\_right\_of\_} \\
 & & \mathsf{pecuniary\_value} \sqcap \mathsf{Documented} \\
 & & \sqcap\ \mathsf{donor}\ \textbf{some}\ \mathsf{Donor} \\
 & & \sqcap\ \mathsf{donee}\ \textbf{some}\ \mathsf{Donee} \\
 & & \sqcap\ \mathsf{duty\_on\_transfer}\ \textbf{some}\ \mathsf{Duty\_on\_gifts} \\
\mathsf{GC\_11\_1c\_F} & \sqsubseteq & \mathsf{Generic\_Case} \sqcap \mathsf{disallowed\_by}\ \textbf{value}\ \textit{section11\_1c} \\
 & \equiv & \mathsf{gratuitous\_creation\_surrender\_or\_waiver\_of\_a\_right\_of\_} \\
 & & \mathsf{pecuniary\_value} \sqcap \mathsf{Documented} \\
 & & \sqcap\ \mathsf{donor}\ \textbf{some}\ \mathsf{Donor} \\
 & & \sqcap\ \mathsf{donee}\ \textbf{some}\ \mathsf{Donee} \\
\mathsf{11\_1c\_Obligation} & \sqsubseteq & \mathsf{Obligation} \sqcap \mathsf{disallows}\ \textbf{only}\ \mathsf{GC\_11\_1c\_F} \\
 & & \sqcap\ \mathsf{allows}\ \textbf{only}\ \mathsf{GC\_11\_1c\_P} \\
 & \equiv & \{\textit{section11\_1c}\}
\end{array}
$$

Section 11-1c expresses an obligation where the default situation is allowed. Therefore we partitioned the case into something that is permitted (complying with the obligation) and something that is disallowed (not complying with the obligation).

Applying norms As our norms are built up from classes in OWL and the case description is specified in terms of individuals, the match between a GC and individual case is through realisation, a specific form of classification. A case is matched when the classifier finds that some part of the case description can be classified as one of the conditions of the GC of some norm. This classification includes all 'expansions' that can be applied to the case description. So in our example, 'donation of copyright' is according to the domain ontology a kind of 'right-of-pecuniary-value'. Norm matching and identifying case terms are part of the same classification process. The *Donee* role also provides an example of the use of the ontology during practising normative assessment. The use case states that we have *donee(donation_of_copyright,mary)*. The reasoner will use the ontology to infer that *mary* has to be of class *Donee*, since the object property *donee* has the class *Donee* as its range. Furthermore, if we specify in the ontology that a *Donee* has to be a *Natural_person*, the reasoner will also infer that *mary* is a *Natural_person*.

We make the distinction between a (deontic) qualification that qualifies a norm and the qualification a case gets – whether generic or individual. The norm is qualified using the three deontic notions that were already discussed. An individual case can only get one out of two possible qualifications, namely *allowed* and *disallowed*. This all has to do with looking at a case from the perspective of its effect on the default and the partitioning of the deontic notions prohibition and obligation into something that is either allowed or disallowed.

Dealing with exceptions GC's themselves can also form class hierarchies. Because the norms are expressed in OWL, Pellet can be used to generate the desired subsumption hierarchy. A more specific case GC_2 will be subsumed by GC_1. This hierarchy actually represents the existing exception relations between the norms, similar to the principle of lex specialis in law. Moreover we can apply the following role inclusion axioms to make the exception relations explicit:

$$\begin{aligned} \mathsf{allows} \circ \mathsf{disallowed\_by} &\rightarrow \mathsf{exception} \\ \mathsf{disallows} \circ \mathsf{allowed\_by} &\rightarrow \mathsf{exception} \end{aligned}$$

This explicit exception property indicates the fact that there is in principle a conflict, as the deontic qualifiers that are associated with the GCs are different ones. OWL semantics is monotonic, it cannot handle such inconsistencies directly. The inferred exception hierarchy is used by the plugin described in Section 3.4 to resolve such possible conflicts. If an individual is both Allowed and Disallowed, the plugin tries to resolve this conflict by checking whether one of the norms allowing or disallowing it contains an exception relationship with the other norm.

We will now explain an example of such an exception in terms of our usecase. According to the regulations the donation specified in our usecase is disallowed by section 11-1c, because it is subject to duty payment, but the individual case does not mention any payment being made. However the donation is also allowed by section 11-3 because it is retained by the donor. In this case section 11-3 is an exception to what is expressed by section 11-1c. It indicates certain conditions under which the obligation expressed by *11-1c* does not hold. This other relevant part for our usecase, Section 11-3, is modeled as follows:

$$\begin{aligned} \mathsf{GC\_11\_3} &\sqsubseteq \mathsf{Generic\_Case} \sqcap \mathsf{disallowed\_by}\ \textbf{value}\ \mathit{section11\_3} \\ &\equiv (\mathsf{gratuitous\_creation\_surrender\_or\_waiver\_of\_a\_right\_of\_} \\ &\quad \mathsf{pecuniary\_value} \sqcap \mathsf{Documented} \\ &\quad \sqcap \mathsf{donor}\ \textbf{some}\ \mathsf{Donor} \\ &\quad \sqcap \mathsf{donee}\ \textbf{some}\ \mathsf{Donee} \\ &\quad \sqcap \mathsf{object\_of\_transfer}\ \textbf{some}\ \mathsf{right\_of\_pecuniary\_value} \\ &\quad \sqcap \mathsf{retained\_by}\ \textbf{some}\ \mathsf{Person} \\ \mathsf{11\_3\_Prohibition} &\sqsubseteq \mathsf{Prohibition} \sqcap \mathsf{disallows}\ \textbf{only}\ \mathsf{GC\_11\_3} \\ &\equiv \{\mathit{section11\_3}\} \end{aligned}$$

The expected verdict of the plugin with regard to the donation from Phil to Mary is that the donation is *allowed*, as article *11-3* is more specific than article *11-1c*: the lex specialis principle states that *11-3* trumps *11-1c*.

3.2 Problems in identity and identifying individuals

3.2.1 A general introduction to the problem

The problem addressed is the fact that the T-Box of OWL, i.e. the class description cannot sufficiently constrain the uniqueness (identity) of individuals in the A-Box.

The fact that OWL does not make a unique naming assumption makes the problem worse. Note that the fact that OWL does not make this assumption is very reasonable in a world where many words mean different things and many different things have the same name. The SW world is such a world. Below we will describe what kind of approaches may provide solutions: approximative or partial (i.e. for certain constructs). In [Hoekstra and Breuker, 2008] the 'deeper' cause of the problem is described (and approximate solutions are proposed). The context of the problem is slightly different from the legal reasoning perspective presented here. In the paper, the the context is 'design patterns'. Design patterns are what is called in Estrella: frameworks, [Breuker et al., 2007] Here we present a short account of the problem as described in the paper.

The problem explained in terms of a regulation

The identity problem boils down to the fact that in some cases we would like to talk about things at the level of individuals, instead of at the level of concepts, which are actually sets of individuals. For example when we want to express a student that checks out a book belonging to a course *he* is enrolled in, this is already extremely difficult - if not impossible - to express in OWL-DL. The problem is getting OWL to understand that the course which the course book belongs to, is the same course that the student is enrolled in.

When using a rule formalism we can easily model this example just by using variables each time we want to point to a certain student or course. Variables are however not present in OWL-DL. Therefore we need to find out in what occasions we really need variables and try and mimic this use of variables as much as possible. After the expermenting with various approaches is finished, the results will be added to this deliverable as well. The various approaches we are using at the moment do not exclude each other. We might be able to combine them to come to an optimal solution.

Section 11-3 of our example demonstrates what we call the identity problem of individuals very clearly: we would like to address the fact that the person that donates vs. the one that retains should both refer to the same individual:

Norm/Generic case Section 11-3

$$\begin{aligned}
\mathsf{GC\_11\_3} &\sqsubseteq \mathsf{Generic\_Case} \sqcap \mathsf{disallowed\_by}\ \textbf{value}\ \textit{section11\_3}\\
&\equiv (\mathsf{gratuitous\_creation\_surrender\_or\_waiver\_of\_a\_right\_of\_}\\
&\quad \mathsf{pecuniary\_value} \sqcap \mathsf{Documented}\\
&\quad \sqcap \mathsf{donor}\ \textbf{some}\ \mathsf{Donor}\\
&\quad \sqcap \mathsf{donee}\ \textbf{some}\ \mathsf{Donee}\\
&\quad \sqcap \mathsf{object\_of\_transfer}\ \textbf{some}\ \mathsf{right\_of\_pecuniary\_value}\\
&\quad \sqcap \mathsf{retained\_by}\ \textbf{some}\ \mathsf{Person}\\
\mathsf{11\_3\_Prohibition} &\sqsubseteq \mathsf{Prohibition} \sqcap \mathsf{disallows}\ \textbf{only}\ \mathsf{GC\_11\_3}\\
&\equiv \{\textit{section11\_3}\}
\end{aligned}$$

The problem can be generalized by the tree-model property of OWL DL: only tree-like axioms are allowed, except for nominals, transitive properties and role inclusion axioms of OWL 2. Complex structures cannot be described precisely due to the lack of predicates with arbitrary number of arguments or circles in axioms.

3.3 Using conjunctive queries

Using OWL as a knowledge representation language can be very effective but users still have to trade expressiveness for decidability and tractable inferences. Two disturbing limitations of the language are the tree-model property and the lack of n-ary predicates. Users familiar with rule formalisms tend to use rules or other extensions supporting variables to overcome these limitations, however, they lose decidable satisfiability checking w.r.t. the T-Box.

When modeling law in the HARNESS architecture using OWL, the above mentioned limitations also emerge. We will present an extension, however, which will solve some of these issues: using *conjunctive queries* for specifying generic cases.

3.3.1 Definition and usage

Conjunctive queries (CQ) are well known in database systems and are in the focus of DL research since years but not yet widely available in OWL applications. Practical results for complex DL languages have only appeared recently [Glimm, 2007]. A possible syntax for such queries has just been defined in SPARQL-DL [Sirin and Parsia, 2007].

A conjunctive query is a conjunction of concept expressions of the form $C(t)$ and role expressions of the form $r(t, t')$ where C is a concept, r is a role and t, t' are terms, i.e., variables or individual names [Glimm, 2007]. All variables are existentially qualified. These conditions are very similar to the body (condition) of SWRL rules. Introducing variables when specifying generic cases basically solves three different issues:

- We are no longer limited by the tree-model property of OWL, generic cases can express arbitrary relational structures.
- In a query, the values for variables can point on the case, or part-of-case the norm refers to. We can keep track of individuals when identifying obligations, permissions and violations.
- Easier modeling: most knowledge experts are familiar with variables and it is easier for them to specify the condition with conjunctive queries. When the expressivity is not required, the CQ can be transformed into an OWL class expression [Gasse and Haarslev, 2008].

The following example demonstrates the usage of CQs. In Section 16. paragraph (2) in the Hungarian Law on Duties (Act XCIII of 1990) an exemption is specified for paying duties on a land received as a gift:

> “In order to verify completion of the construction of the residential house referred to in Paragraph g) of Subsection (1) the state tax authority

shall contact the competent building authority within 15 days following the expiry of the 4-year time limit specified therein. If the building authority provides a certificate in proof of the occupancy permit issued to the name of the property owner, the state tax authority shall cancel the duty assessed, but suspended in respect of payment."

The condition for the exemption can be formalized using conjunctive queries the following way. If the gift ($?g$) is a *plot of land*, and a building ($?b$) is built on it, which is a *residential house*, and an *occupancy permit* ($?p$) was issued to the name of the *donee* ($?d$), the generic case is fulfilled:

$$\begin{aligned} GC_{S16\_2} \equiv\ & Donation(?t) \wedge donee(?t, ?d) \\ & \wedge subject(?t, ?g) \wedge PlotOfLand(?g) \\ & \wedge built\_onto(?b, ?g) \wedge ResidentialHouse(?b) \\ & \wedge permit\_issued(?b, ?p) \wedge OccupancyPermit(?p) \\ & \wedge issued\_to(?p, ?d) \end{aligned} \tag{3.1}$$

A conjunctive query can be represented by a graph where each variable in the query give rise to a node in the graph. Concept names appear as node labels, role names as edges at the appropriate variables in the graph. Figure 3.2 shows the graph for the example CQ.

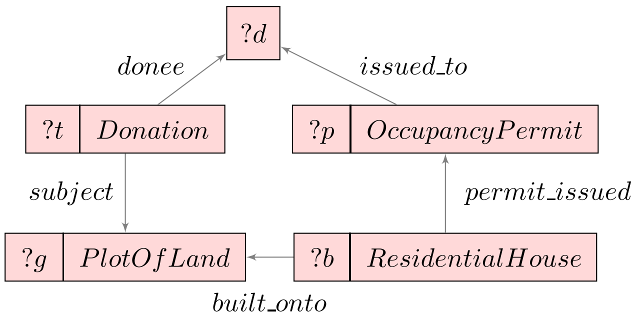


Figure 3.2: Graph representation of a conjunctive query

3.3.2 Inferences with conjunctive queries

Similar to OWL classes different inference services can be implemented for conjunctive queries:

query entailment is the decision problem to answer whether a query is true in all models of a knowledge base, whereas

query answering is the problem of finding all answer tuples for a query. If the entailment is false, there are no answers. Otherwise there may be one or more tuples which fulfill the constraints described in a query.

satisfiability is to decide if a knowledge base has at least one model in which the query is true. When building a model and respective queries, a non-satisfiable query may indicate inconsistency in the model, as the corresponding legal condition will never be fulfilled.

subsumption of CQs can be defined in a similar manner as class subsumption: with respect to a knowledge base $\mathcal{K}$ a query Q_1 subsumes the query Q_2 if in all models where Q_2 is true, Q_1 is also true.

Not all of these problems are solved for the description logic underlying OWL 2. Satisfiability can be easily answered using a DL reasoner, but it is still an open issue whether the other problems are decidable for the DL $\mathcal{SHOIN}(\mathcal{D})$. Latest results showed that query entailment is decidable for $\mathcal{SHIQ}$ [Glimm et al., 2007a] and $\mathcal{SHOQ}$ [Glimm et al., 2007b] which are slightly restricted sublanguages of OWL 2.

However in the general interpretation variables in a CQ are not required to correspond to some named individual in the ABox. For so-called *non-distinguished variables* only the existence of a suitable element is required in the model, and *answer variables* are required to have a corresponding named individual. This is important as in the restricted closed-world interpretation of CQs we only use answer variables, and then all inference problems are decidable in OWL 2.

A subsumption hierarchy of CQs can be derived the same way as for OWL classes. Conjunctive queries are a generalization of OWL class expressions, as all class expression can be trivially transformed to an atomic CQ with one variable:

$$C \to Q(x) \equiv C(x)$$

As a result subsumption can be defined across CQs and OWL named classes, and hierarchy of CQs and classes can be merged. As an example for the CQ in equation 3.1: $GC_{S16\_2} \sqsubseteq Donation$.

Satisfiability of conjunctive queries can be derived from subsumption the same way as for OWL classes, by defining the always unsatisfiable CQ:

$$Q_\perp(x) \equiv \perp(x)$$

$$Q(\dots) \text{ is unsatisfiable} \Leftrightarrow Q(\dots) \sqsubseteq Q_\perp(x)$$

3.3.3 Application in HARNESS

In HARNESS we have two distinct modeling issues: creating a domain model for enforcing valid case descriptions and legal assessment. In the assessment part we would provide definitions for generic cases (GC). The domain model must be consistent and all kind of inferences are required. With GC descriptions, however, the only inferences required are hierarchy of GCs (is one description more general than another?) and case entailment (does a case fulfill all conditions of a GC?), as introduced in Section 2.2.

We can use conjunctive queries to describe generic cases while the rest of the model is still specified in OWL2. As generic cases describe a selection for all cases when norms are interpreted, it is natural to use a query language for this purpose. Conjunctive queries are appropriate because inference services are available to cover the tasks required in HARNESS:

- query answering to match the case with norms,
- subsumption relations to extract exceptions (lex specialis) for norms in the model and
- satisfiability to ensure model consistency.

An experimental implementation for the closed-world interpretation (using only answer variables) is provided based on the Pellet DL reasoner.

3.4 A GUI for prototyping HARNESS

For the test environment we are developing a Protégé 4 plugin, which should allow the user to select an ontology containing norms and an ontology containing a case description. A third, standardized, ontology containing definitions of what norms actually are should also be in the imports closure of the other two ontologies. The test environment should be able to classify the ontologies and determine whether the individual cases are permitted, obliged or prohibited. The user should also be given the option to ask for an explanation, that is, why a certain individual case is one of those three options.

The program The program as it exists now offers the option to select two OWL ontologies, one containing the norms and one containing the case description(s). When done, the user can press a button causing the program to classify and realise the ontologies using the pellet reasoner[2]. If the earlier mentioned standardized ontology containing definitions of what norms are is not present in the imports closure of the ontologies, the program will give an error and will not proceed. If it is present, the program will do the following:

First, it will generate a list of all individuals in the case description ontology, and then it will ask the reasoner which properties each individual has. If one of them is *allowed_by* or *disallowed_by*, the program places the individual in a list for the user to see, and in another list which also stores the norm (represented by another individual) by which it is (dis)allowed.

After the program is done classifying and realizing the ontologies and checking the individuals as described above, the end result should be two lists, one with 'allowed individuals' and one with 'disallowed individuals'. The user can then get an explanation why an individual is allowed or disallowed by clicking on it in one of the lists. For the explanation, we query pellet directly. The pellet API offers two methods for getting the explanation; one that pretty-prints it and one that simply returns a list of OWL axioms. The program prints both versions to the screen.

The program, however, is still incomplete. It still has to generate a final verdict, that is, determine whether the individual cases in the case descriptions are permitted,

[2]http://pellet.owldl.com/

obliged or prohibited. Right now, the program is not yet able to determine if an individual is obliged. In the following two simple situations, it is able to determine whether it is permitted or prohibited:

1. If a certain individual only has the *allowed_by* property and not the *disallowed_by* property, the verdict is *Permitted.* The other way around, the verdict would be *Prohibited.*

2. If an individual has both properties, we try to resolve this by checking if one of the norms allowing or disallowing it is in a subsumption relation with the other. If so, we can conclude that the norm subsumed by the other is more specific, and therefore the one that should be applied (the so-called *lex specialis* principle).

To determine if a norm (again, represented by an individual) is subsumed by another, for example the norm allowing the case, we look at the type of the norm (individual) disallowing the case (which is a class), collect all ancestor classes of that class, and check if all classes which are types of the individuals allowing the case, are in that list of ancestor classes. If so, we can safely conclude that the norm allowing the case is more specific than the ones disallowing it, and the verdict is *Permitted.* The other way around, the verdict would be *Prohibited* again.

Once a verdict is generated, the individual is once again placed in a list for the user to see. The user can again ask for an explanation, but this time we do not use Pellet to generate it but rather we have written a small explanation function ourselves, which basically comes down to explaining the above and showing the class hierarchy.

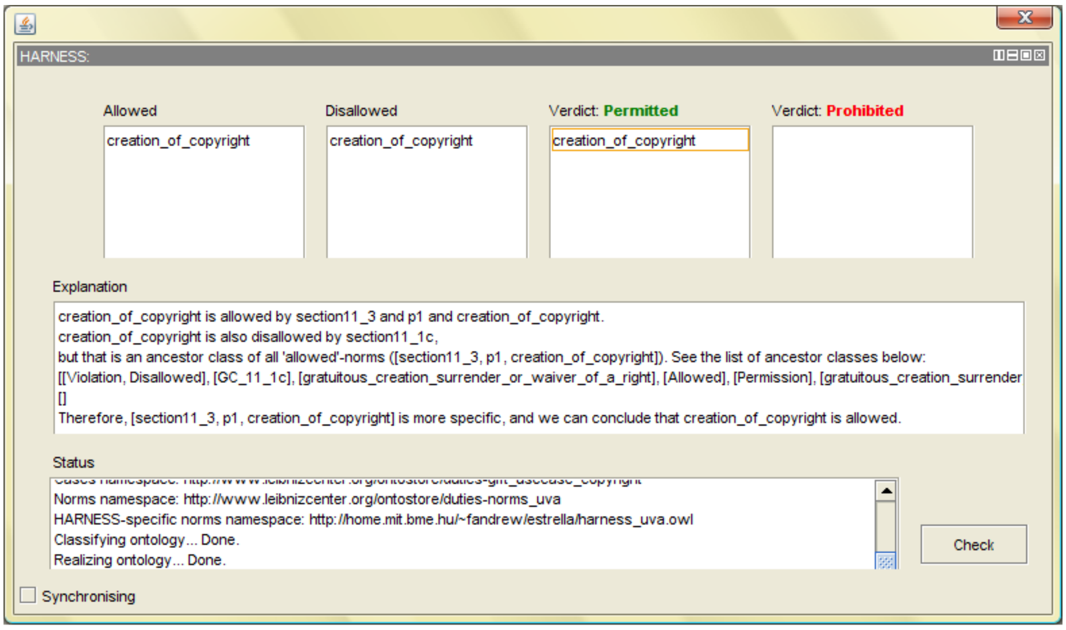
The implementation is done in Java, which is a natural choice given the APIs that exist for both OWL ontologies and the Pellet reasoner[3].

In Section 3 we explained our usecase which describes a donation of a copyright. As is shown in the screenshot, the final verdict of OWL Judge concerning this usecase is indeed that the individual *creation_of_copyright* is disallowed by *11-3*, but allowed by *11-1c*. The explanation indicates that 11-3 is the most specific norm, which makes the final verdict "Permitted".

The test environment as it exists now implements the basic requirements we have for it, but there are still some extensions that we can consider. For instance, it might be possible to enhance the explanation by allowing the user to click on a certain individual case to get more information about it. More specifically, as an example, in the screenshot (Figure 3.3) you can see that the individual *amy* is disallowed by *art1c*. That information in itself is not of much use yet, unless of course you can click on *art1c*, thus getting more information about it.

As mentioned, the program right now is not yet capable of determining whether an individual case is *Obliged*, which is also something that needs to be dealt with. Furthermore, we need to look at a way to further enhance the explanations, since the explanations given to us by Pellet are not always convincing.

[3]http://owlapi.sourceforge.net



Allowed: creation_of_copyright

Disallowed: creation_of_copyright

Verdict: **Permitted**: creation_of_copyright

Verdict: **Prohibited**:

Explanation

```
creation_of_copyright is allowed by section11_3 and p1 and creation_of_copyright.
creation_of_copyright is also disallowed by section11_1c,
but that is an ancestor class of all 'allowed'-norms ([section11_3, p1, creation_of_copyright]). See the list of ancestor classes below:
[[Violation, Disallowed], [GC_11_1c], [gratuitous_creation_surrender_or_waiver_of_a_right], [Allowed], [Permission], [gratuitous_creation_surrender
[]
Therefore, [section11_3, p1, creation_of_copyright] is more specific, and we can conclude that creation_of_copyright is allowed.
```

Status

```
Norms namespace: http://www.leibnizcenter.org/ontostore/duties-norms_uva
HARNESS-specific norms namespace: http://home.mit.bme.hu/~fandrew/estrella/harness_uva.owl
Classifying ontology... Done.
Realizing ontology... Done.
```

Figure 3.3: Screenshot of the Protégé plugin

3.5 An experimental test domain: a library regulation

Here, an experimental test domain is given, that highlights some of the intricate problems surfaced in experimenting with the (deontic) reasoning capabilities of OWL-DL. The document is largely based on and complements the university library regulations ontology developed at the University of Amsterdam (see/open `usecase_amy.owl`); it consists of the following: provision of the university library regulations source text, use cases constructed from this regulation, how this regulation differs from *general* legal regulations, a worked out example of how the norms in the regulations were modeled in OWL-DL and how they pertain to the given example, and finally a description and an example of the problem of identity as it pertains to the university library regulations . Each of these will be addressed below.

University Library Regulations As mentioned earlier, the university library regulations domain was constructed in order to surface the limits of (deontic) reasoning in OWL-DL. The following (small) domain can be understood as an experimental testbed that provides sufficient insight into the limitations as well as the advantages of using an ontology-based approach to legal knowledge representation and the utilization of an existing OWL-DL reasoner, namely, Pellet. There are both merits and inadequacies in using such a small domain: on the one hand, a small domain makes it easier for the modeler to spot the different problems that arise in reasoning over the modeled knowledge. More so, a small domain is highly manageable, does not require too much time to construct, and can provide sufficient insight into the different capabilities of OWL-DL, given its current expressivity.

On the other hand, a small domain hardly resembles a real-world example of legislation. This is potentially a serious problem, since it remains uncertain whether the insights gained from a small experimental domain are on par with the kinds of issues surfaced from a real-world legal domain. Nevertheless, as stated previously, the smaller experimental domain is meant to be more manageable, and attempts at surfacing possible *intrinsic* limitations in knowledge representation formalisms, such as OWL-DL. Later, it will be shown how this domain in particular differs in an important aspect from most legal regulations - an issue independent of the formalism employed for representing a given domain.

The norms (or articles) in the university library regulations are as follows:

1a) Only students registered at this university are allowed to check out a book from this library.

1b) Students registered at other universities are allowed to check out a book from this library provided that they are enrolled in at least one course given at this university.

1c) Students who have checked out more than five books are not allowed to check out another book.

2a) Students registered at this university are not allowed to check out books other than those belonging to the courses they are enrolled in.

2b) Students registered at this university, with blue hair, are allowed to check out books not belonging to a course they are enrolled in.

2c) Students registered at this university who inadvertently checked out a book , should return the book forthwith or dye their hair blue.

By closely inspecting each of the above articles, it can be seen that the above rules form an intuitive heirarchic pattern - one of subsumption relationships. To illustrate, consider the following:

- $\textsc{Art1b} \sqsubseteq \textsc{Art1a}$

  Rule 1b is an exception to 1a, as 1a implicitly expresses that students registered at other universities are not allowed to check out books if they are not enrolled in at least one course at this university. Because it is an implicit exception, it is not derived explicitly. However as the default specifies that everything is by default disallowed unless stated otherwise, the correct behavior will be inferred anyway.

- $\textsc{Art1c} \sqsubseteq \textsc{Art1a}$ and $\textsc{Art1c} \sqsubseteq \textsc{Art1b}$

  While Art.1c is an exception to 1a, it is also an exception to Art.1b. This can be remedied by imposing a cardinality restriction on the amount of books that can be checked out. In so doing, Art.1a and Art.1b can be sufficiently differentiated.

- $\textsc{Art2b} \sqsubseteq \textsc{Art2c}$

- $\textsc{Art2c} \sqsubseteq \textsc{Art2a}$

Use Cases In order to verify whether each of the respective articles (norms) were modeled correctly, a set of use cases were constructed. The desired inferential behavior with respect to the norms is shown below (e.g. allowed by art1a):

1. A student registered at this university that checks out a book → allowed by art1a

2. A student registered at a different university that checks out a book and is enrolled in one course given at this university → allowed by art1b

3. A student registered at a different university that checks out a book and is not enrolled in a course given at this university → disallowed by defaultnorm

4. A student (registered at this university or not) who has checked out more than 5 books → allowed by art1a, disallowed by art1c

5. A student registered at this university that checks out a book not belonging to a course he is enrolled in → allowed by 1a, disallowed by art2a

6. A student registered at this university that checks out a book belonging to a course he is enrolled in → allowed by art1a

7. A student registered at this university with blue hair that checks out a book (not belonging to a course he is enrolled in) → disallowed by art2a, allowed by art1a, art2b and art2c

8. A student registered at this university that checks out a book not belonging to a course he is enrolled in and returns the book → disallowed by art2a, allowed by art1a and art2c

Default Normative World The university library regulations differs from real-world legislation in an important way: specification of the default normative ruling as allowed or disallowed. This principle with respect to the library regulations translates into an ontological viewpoint where everything is intially disallowed, unless explicity stated by the law to be otherwise. For the library regulations then, the default prescription is that checking out books is disallowed. In accordance with what the default world specifies, the norms have to be modeled so as to account for this prohibitive default situation, thereby addressing how different normative behavior (i.e., the generic cases) is reflected in accordance with the deontic operators.

To exemplify the current modeling approach, consider the following example: a student, Amy, who is registered at the University of Amsterdam has decided to check out 6 books for a thesis she has to write. Amy's case can be represented (in OWL-DL) by the following:

$$\begin{aligned} \texttt{Registered\_Student}(Amy) &\wedge \texttt{registered\_at}(Amy, UvA) \\ \texttt{checks\_out}(Amy &, [general\_book\_1, general\_book\_2, \\ &\quad general\_book\_3, general\_book\_4, \\ &\quad general\_book\_5, general\_book\_6]) \end{aligned}$$

According to the regulations Amy's situation is allowed by article *1a*, but disallowed by article *1c*. However, these two articles contain an intuitive hierarchic pattern: the case described by article *1a* subsumes that of article *1c*. In fact, article *1c* is an exception to *1a*, because *1a* implicitly expresses that students registered at *other* universities are not allowed to check out any books. The expected verdict of the system should therefore be that Amy's situation is *disallowed*, as article *1c* is more specific than article *1a*, the lex specialis principle states that *1c* trumps *1a*.

Although the two relevant articles are article *1a* and *1c*, we first need to specify the default situation. In the library regulations the default situation is disallowed; students are generally not allowed to check out books, unless stated differently in the norms. The default norm and generic case apply to all situations where a book is checked out:

$$\begin{aligned} \mathsf{Default\_GC} &\sqsubseteq \mathsf{Generic\_Case} \sqcap \mathsf{disallowed\_by}\ \textbf{value}\ \mathit{defaultnorm} \\ &\equiv \mathsf{checks\_out}\ \textbf{some}\ \mathsf{Library\_Book} \\ \mathsf{Default\_Norm} &\sqsubseteq \mathsf{Prohibition} \sqcap \mathsf{disallows}\ \textbf{only}\ \mathsf{Default\_GC} \\ &\equiv \{\mathit{defaultnorm}\} \end{aligned}$$

Article *1a* states that students registered at this university are allowed to check out a book from this library:

$$\begin{aligned} \mathsf{Art1a\_GC} &\sqsubseteq \mathsf{Generic\_Case} \sqcap \mathsf{allowed\_by}\ \textbf{value}\ \mathit{art1a} \\ &\equiv \mathsf{Registered\_Student} \sqcap \mathsf{checks\_out}\ \textbf{some}\ \mathsf{Library\_Book} \\ \mathsf{Art1a\_Permission} &\sqsubseteq \mathsf{Permission} \sqcap \mathsf{allows}\ \textbf{only}\ \mathsf{Art1a\_GC} \\ &\equiv \{\mathit{art1a}\} \end{aligned}$$

Article *1c* states that students who have checked out more than five books are not allowed to check out another book:

$$\begin{aligned}
\mathsf{Art1c\_GC\_F} &\sqsubseteq \mathsf{Generic\_Case} \sqcap \mathsf{disallowed\_by}\ \textbf{value}\ \mathit{art1c} \\
&\equiv \mathsf{Student} \sqcap \mathsf{checks\_out}\ \textbf{min}\ 6
\end{aligned}$$

$$\begin{aligned}
\mathsf{Art1c\_GC\_P} &\sqsubseteq \mathsf{Generic\_Case} \sqcap \mathsf{allowed\_by}\ \textbf{value}\ \mathit{art1c} \\
&\equiv \mathsf{Student} \sqcap \mathsf{checks\_out}\ \textbf{some}\ \mathsf{Library\_Book} \\
&\quad \sqcap \mathsf{checks\_out}\ \textbf{max}\ 5
\end{aligned}$$

$$\begin{aligned}
\mathsf{Art1c\_Prohibition} &\sqsubseteq \mathsf{Prohibition} \sqcap \mathsf{disallows}\ \textbf{only}\ \mathsf{Art1c\_GC\_F} \\
&\quad \sqcap \mathsf{allows}\ \textbf{only}\ \mathsf{Art1c\_GC\_P} \\
&\equiv \{\mathit{art1c}\}
\end{aligned}$$

Article *1c* expresses a prohibition where the default situation is disallowed. The GC is therefore partitioned into a part that is disallowed (the prohibition) and the thing that is allowed (the context). In this case it is permitted to check out books, checking out more than 5 books is prohibited. By using a cardinality restriction, checking out less than or exactly 5 books is allowed by this rule.

The individual *art1c*, which represents the prohibition from article *1c*, disallows our individual *amy*, since her situation matches with Art1c_GC. However, *amy* is also classified as Art1a_GC, and therefore *amy* is also allowed_by *art1a*. In this way we can derive the exception relation between *art1c* and *art1a*:

$$\mathsf{exception}(\mathit{art1c}, \mathit{art1a})$$

Finally it is interesting to take a look at article 2a, which is modeled as follows:

$$\begin{aligned}
\mathsf{Art2a\_GC} &\sqsubseteq \mathsf{Generic\_Case} \sqcap \mathsf{disallowed\_by}\ \textbf{value}\ \mathit{art2a} \\
&\equiv \mathsf{Registered\_Student} \sqcap \mathsf{checks\_out}\ \textbf{some}\ (\mathsf{Course\_Book} \sqcap \\
&\quad \mathsf{belongs\_to}\ (\mathsf{Course} \sqcap \textbf{not}\ \mathsf{followed\_by}\ \mathsf{Student})) \\
\mathsf{Art2a\_Prohibition} &\sqsubseteq \mathsf{Prohibition} \sqcap \mathsf{disallows}\ \textbf{only}\ \mathsf{Art2a\_GC} \\
&\equiv \{\mathit{art2a}\}
\end{aligned}$$

An interesting question at the time we modeled this norm was: How do we model "books other than those belonging to the courses THEY are enrolled in"? It is possible to model books that belong to a course a student is (not) enrolled in, but to point out a particular student comes down to the identity problem of individuals again. When modeling the articles 2b and 2c, the same problem arises.

Chapter 4

Conclusions, extensions and exploitation

Technical advances and problems

The experimental work on HARNESS presented in the previous chapter covers the two most promising approaches in DL based hybrid normative reasoning. [1]. The most remarkable conclusion that can be drawn is that normative reasoning can be performed with a class-based, monotonic and deductive DL reasoner. This is remarkable, because this approach was not directly pursued - moreover it is a unique one, not only in practice, but also in research. Legal knowledge systems are rule based (often: in Prolog) and from a deontic logic perspective require non-monotonic, 'defeasible' reasoning [2].

This HARNESS solution is deductive and monotonic. Although we have shown the feasibility of the approach it should still be kept in mind that this assessment task plays a minor but essential role in legal problem solving. It can be compared to the use of calculations in engineering. In this analogy, HARNESS can be viewed as a reliable legal 'calculator'.

A less remarkable, but also important finding is that legal ontologies can be used for reasoning about legal cases. This is also unique, as ontologies in legal applications are not really used for reasoning, but have a role in information retrieval [Breuker et al., 2008a]. This is surprising, as OWL has been conceived in the first place for making inferences, but the main use of the reasoning capabilities has been consistency checking and the discovery of implied classes, i.e. in the modeling of ontologies rather than as part of a automated problem solver. Also, in the context of Semantic Web applications in general, use as a knowledge base for reasoning is

[1] About three other alternatives were also investigated, but these appeared less promising: they are not reported here.

[2] One may argue that in fact this non-monotonic moment occurs outside the Pellet reasoner when the 'proto-HARNESS' plug-in selects between two conflicting norms which have an exception relation: the most specific one according to *lex specialis*. However this 'trick' is required because this last step is in fact a meta-level reasoning one: the qualifications 'allowed/disallowed' are *about* a case and not part of a case. If it were the latter, one would need retraction of one of the alternatives to obtain a non-conflicting interpretation of the case. In our solution it is that different sources (norms) say different things about the same (part of) a case. OWL/Pellet does not allow meta-level representations (reification), but via tricks it is possible to have meta-level reasoning mixed with object level reasoning. That is what our first approach also shows: the deontic qualifiers of the norms are in fact meta to the ontology and the Generic Case descriptions. For more technical details see [van de Ven et al., 2008b, van de Ven et al., 2008a]

still rare, and an ongoing topic for research. As has been shown, ontologies work very well in providing a certain level of semantic understanding of a case. Pellet provides a full semantic interpretation of the individuals in the A-Box (the case description). By 'full', it is meant that all inferences are made about properties of individuals on the basis of the definitions in the ontology.

However, a serious limitation in the interpretation of cases remains: it is not always possible to keep track of individuals. For instance, it is not possible that OWL/Pellet will see that in a selling transaction the one who pays – say Tom – is also the receiver of the goods. It will recognize that the payer Tom and receiver Tom have the same properties etc., but it is hard to provide constraints so that no other Tom is the receiver (note that OWL does not make a unique naming assumption). It is even harder when properties of Tom explicitly change due to the transaction: if it is modeled that Tom owns money before the transaction and owns goods after that, OWL cannot identify that Tom is still the same Tom. Also to some extent this can be solved by careful modeling. Being involved in the development of OWL 2, it was known that we were not the first to identify this problem, but have nevertheless proposed interesting approximate solutions [Hoekstra et al., 2006, Hoekstra and Breuker, 2008], having witnessed that the problem also surfaces in legal case descriptions. However, in these explorations the problem has been posed on purpose, but information regarding its prevalence in legal case descriptions is still insufficient, and remains a topic for further research.

Further research

At the end of a research project, ranging from a PhD thesis to a European IP, there are always issues for further research, as is the case here. One important aspect that requires further investigation is the identity of individuals problem. Also, to some extent, the way deontic qualifications work have been simplified. For instance, obligations may have a more complex character when there is a duration between the moment an obligation is applicable and the fulfillment of the obligation. The complexity increases when one takes into account that there are reciprocal relations in legal roles. For example, on the basis of the rule that 'all' vehicles keep to the right (or to the left), expectations that one will not meet a vehicle in the opposite direction under normal circumstances (except, etc.) are deemed legitimate. However, unlike many of these research projects, these problems require serious further investigation.

In the research project AGILE[3] that has recently started, the UvA will build further on HARNESS and the experiences reported here. This project is aimed at developing a methodology for modelling sources of law and at an architecture for legal services in a distributed environment. As the project has also market parties (vendors and other stakeholders) involved, also support tools will be developed for coping with the dynamic nature of legal domains.

[3]Advanced Governance of Information services through Legal Engineering is a 4 year / 16 person year project, financed by the Dutch Organization for Scientific Research (NWO) in which the Leibniz Center for Law of the University of Amsterdam cooperates with the Technical University of Delft.

Towards a hybrid architecture: more R & D

As explained in Chapter 2, our aim was to construct an architecture that covers LKIF completely. Although hybrid architectures have been developed in the past, this ambition had to be abandoned if we wanted to maintain both the expressivity of LKIF-rules and the semantic integrity of OWL-DL. Even at this moment the state-of-the-art in research on the topic of combining a DL-KR such as OWL-DL and a rule formalism are still an important and ongoing topic of research (see e.g.`http://www.w3.org/2005/rules/wiki/RIF_Working_Group`). Still it is maintained that a hybrid solution is required if only for a definite solution for the identity problems. As mentioned earlier, there are two approaches one can adopt. The first one is the way the W3C committee on rules pursues: using DL-safe rules and arriving at a precise definition of what that means. The alternative is to *use* an (expressive) rule formalism in such a way that it cannot tamper with these constraints: for example, in providing 'finishing touches'. This is an area that is not yet sufficiently explored. Moreover, we may return to LKIF-rules and translate some constructs back into the way norms are currently handled in our approach. For instance, the 'unless' construct in LKIF rules becomes superfluous in the specification of the exceptions of norms as OWL/Pellet will discover these constructs automatically.

An architecture is more than some combination of reasoning engines and knowledge bases. It needs to communicate with the user in a meaningful way. Communication with the user is not simply an 'add-on'. Two issues in user communication have been addressed in this report. Aside from getting the information that some case description is allowed or disallowed by some norm(s), one needs a *justification* in terms of the legal source represented by the norms. This issue is extensively researched and developed in WP3, and solutions are readily available. However, to get a focused *explanation* why a norm matches a case, the output from Pellet is far too 'verbose'. Further research on this issue is needed, even if most of these explanations are not complex at all and can be solved by the common sense of the user. More important and complex is the issue of providing an adequate user interface for specifying a case. The problem is that (legally naive) individuals may specify lots of information that is not legally relevant and miss out information that is. The advantage of the classical legal knowledge systems that direct the user fully in the dialogue to obtain the data is that they at least cover legally relevant issues (but may miss out some 'by implicit design'). However such dialogue may turn out to be a Procrustean bed imposed over the 'real' case. Therefore, some method has to be found where the the dialogue can be focused on relevant issues and terms, as well as leaving sufficient room for the user to avoid compliance with a description that is more or less forced upon her. Already in the 80-ies [Clancey, 1983] pointed out that in (rule based) knowledge systems, the dialogue with the user was driven by the needs of the automatic problem solver, often not corresponding to the topic structure in the mind of the user. As architectures have hardly been developed where the problem solver and the dialogue manager could be clearly separated, research in this area is not very well developed.

Market perspectives

The well-founded normative reasoning by this 'proto'-HARNESS certainly has important advantages: in particular due to the fact that Pellet's reasoning is proven to be correct, complete and decidable. Although this is a good result, it is only a small, and even not a direct step into our more ambitious goal of creating an architecture that covers LKIF completely. From this perspective, the current Carneades comes much closer. In this vein, the HARNESS approach is not an alternative to Carneades, but is rather *complementary* to the rule based engines used by vendors of (legal) knowledge systems. HARNESS offers the option to build legal knowledge systems whose reasoning can be trusted blindly. This is not required for any legal knowledge system if the outcomes of the system are not really critical or when the legislation modeled is small and transparent enough to avoid error. For the HARNESS solution, it comes at a price: the modeling takes considerable more effort, even when advanced editing tools are already available. [4] Therefore, we can expect the HARNESS solution – once developed into a practical application – to address a special market. A typical market is the support of legal drafters. Drafting regulations is known to be a costly and long term process that involves careful modeling. Despite this care, most (non trivial) regulations (laws, contracts) contain inconsistencies because no other means are available for checking consitency than running typical cases by 'eye'. Note that the HARNESS approach is not only capable of checking consistency and generating correct solutions, but also in generating all cases that can be distinguished by a regulation: usually that is some orders of magnitude than one can imagine [5].

[4]The typical tools available Protégé (free), TopBraid (commercial) and KAON (commercial) are not only fully developed and widely used, but they are still improved on a rapid rate, making life easier for the user.

[5]A better test set consists of a database of empirically collected cases, for example under a previous version of some legislation, but usually these are not complete and moreover, they are usually not in a format that allows automatic validation. Moreover, a new legislation may refer to cases that did not exist under the previous version.

Appendix: Combining Pellet and SPARQL; classifying with norms

This Appendix is a note written by Szymon Klarman which explains the initial starting point of the use of OWL to represent and reason with norms. SPARQL, the query language for RDF and OWL, is used here instead of 'rules' to maintain identity of individuals and to allow for meta-level conclusions.

SPARQL can be used to express queries across diverse data sources, whether the data is stored natively as RDF or viewed as RDF via middleware. SPARQL contains capabilities for querying required and optional graph patterns along with their conjunctions and disjunctions. SPARQL also supports extensible value testing and constraining queries by source RDF graph. The results of SPARQL queries can be results sets or RDF graphs."[6] OWL ontology can be represented as an RDF graph and therefore queried using SPARQL (through Jena reasoner), which in turn can trigger generation of new triples via CONSTRUCT expressions. In this sense SPARQL can sufficiently mimic a rule-like behavior over ontologies. In principle, however, an OWL graph will not contain all the information that is logically entailed by the knowledge base. This limitation can be to a large extent overcome by referring to an *RDF inference graph* instead, as generated by an OWL reasoner such as Pellet. The essential reasoning services required by HARNESS can be provided on a satisfactory (i.e. practical) level by combining the functionalities of Pellet and SPARQL, given that:

- a proper interaction between an ontology, Pellet's reasoning and SPARQL queries is assured,
- the reasoning process is supported by interaction with the user, and managed by an external program.

It should be clear from the beginning that the proposed solution is a compromise between the assumed requirements and the available formalisms/tools, and has several limitations, which can affect some of the logical properties of the reasoning procedure. These will be pointed out below.

Two clear, general advantages of the approach are:

[6] `http://www.w3.org/TR/rdf-sparql-query/`.

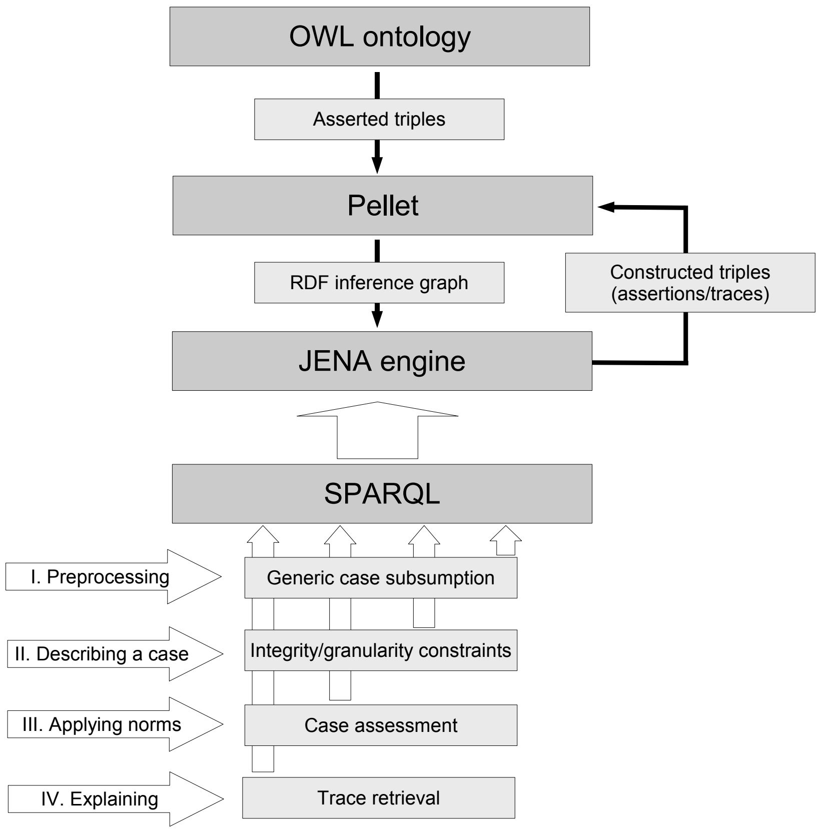


Figure 4.1: The reasoning architecture and the relevant reasoning tasks.

- Compliance to Semantic Web standards (SPARQL has W3C recommendation status).
- A good position for improving the system in the future, due to on-going efforts on tightening the co-operability of Pellet – SPARQL and SPARQL – OWL (cf. [Sirin and Parsia, 2006], [Parsia and Sirin, 2007], [Kremen and E.Sirin, 2008]).

Overview An RDF inference graph contains the results of *classification*, i.e. subsumption hierarchy of named classes, and *realization*, i.e. assignment of *named individuals* to *named classes* and *pairs of named individuals* to *relations*. Jena engine allows, moreover, to access blank nodes representing inferred individuals, i.e. individuals entailed by existential restrictions. Finally, we can query about the knowledge explicitly asserted in the TBox and ABox of an ontology. Properly controlled, SPARQL can be used to provide support for the following tasks (see figure 4.1):

1. Generating the *hierarchy of generic case descriptions* required for detecting exceptions in norms and resolving normative conflicts.

2. Supporting the process of describing individual cases, by imposing additional *integrity constraints* that are not expressible in OWL (frameworks) or *granularity constraints*, i.e. requirements for the level of specificity of information provided by the user.

3. Performing *case assessment*, i.e. applying norms to individual cases described by user.

4. *Retrieving information* necessary for formulating *relevant explanations*.

The basic idea (already incorporated in TopBraid) is to iterate between Pellet and Jena engine so that all results of Pellet's reasoning are passed on to Jena, and all new triples generated by Jena via SPARQL constructs are fed back to Pellet, as long as no new inferences are obtained. Since we intend to address several tasks, which should be performed incrementally, the interaction between reasoners, SPARQL queries, and the user should be managed by an external program. Moreover, during the process we can use RDF for leaving traces of crucial inference steps triggered by SPARQL, which could be useful in further stages.

In the following sections the potential support for the respective tasks is evaluated[7].

Reasoning Tasks

Generic case subsumption A generic case is a conjunction of positive and/or negative literals, consisting of concept/role names present in the ontology, variables, and possibly some constants (individual names also belonging to the ontology).

For short, we will comply here to the first of the possible representations of generic cases, described in the next section. Hence we will represent a generic case as the following concept definition:

$$\textsc{Case}_i(y) \equiv \varphi(\overline{x}) \bigwedge_{x\in\overline{x}} \mathsf{part\_of\_case}(x, y)$$

meaning that if there exists a ground substitution of the sequence of variables $\overline{x} = x_1, ..., x_n$ satisfying the condition $\varphi(\overline{x})$ and all substituted individuals are $\mathsf{part\_of\_case}$ y then y is an instance of a generic $\textsc{Case}_i$. We say that a generic case $\textsc{Case}_2$ subsumes a generic case $\textsc{Case}_1$ (given a background knowledge base $\mathcal{K}$) if the following axiom holds in $\mathcal{K}$:

$$\textsc{Case}_1 \sqsubseteq \textsc{Case}_2$$

This simply means that $\textsc{Case}_1$ is more specific than $\textsc{Case}_2$ and so any individual case satisfying conditions of $\textsc{Case}_1$ has to satisfy the conditions of $\textsc{Case}_2$. Given two case descriptions:

[7]DISCLAIMER: All examples of SPARQL constructs included below are sketchy and in many cases may be incomplete and syntactically incorrect. A thorough revision is necessary.

$$\textsc{Case}_1(y) \equiv \varphi(\overline{x}) \bigwedge_{x \in \overline{x}} \mathsf{part\_of\_case}(x, y)$$
$$\textsc{Case}_2(y) \equiv \psi(\overline{x}) \bigwedge_{x \in \overline{x}} \mathsf{part\_of\_case}(x, y)$$

we can determine whether $\textsc{Case}_2$ subsumes $\textsc{Case}_1$ if the following formula is inconsistent with $\mathcal{K}$:

$$\textsc{Case}_1 \sqcap \neg\textsc{Case}_2$$

This in turn can be verified if for any arbitrary ground substitution satisfying:

$$\varphi(\overline{x}) \bigwedge_{x \in \overline{x}} \mathsf{part\_of\_case}(x, a)$$

it can be proved that:

$$\forall_{\overline{x}} \neg(\psi(\overline{x}) \bigwedge_{x \in \overline{x}} \mathsf{part\_of\_case}(x, a))$$

This reasoning is computationally hard (possibly undecidable), since it requires checking all possible models. Nevertheless, under some assumptions the task can be simplified and resolved with SPARQL. For this purpose we need to use a knowledge base with an empty ABox (excluding individuals that comprise a fixed part of the ontology, e.g. European countries in the model of the tax directive), and run the following algorithm for every generic case $\textsc{Case}_j$:

1. Generate an arbitrary, minimal instantiation of the condition of $\textsc{Case}_j$, i.e. describe an arbitrary individual case a.

2. Run the case assessment applying all generic cases.

3. Retrieve all generic cases $\textsc{Case}_i$ such that a is classified as an instance of $\textsc{Case}_i$.

4. For every $\textsc{Case}_i$ create a TBox axiom $\textsc{Case}_j \sqsubseteq \textsc{Case}_i$.

5. Remove all created individuals and assertions.

One source of problems can stem from the shift from monotonic to nonmonotonic semantics required for expressing negation-as-failure. Notice, that whereas we can use classical negation for unary predicates (be referring to complements of classes), it is practically impossible to obtain the same effect for roles. Therefore the only form of negation that can be used with respect to roles is NAF. Consider two generic cases 1) $\varphi = C(x)$ and 2) $\psi = C(x) \wedge \neg r(x, y)$. The procedure will infer that case 2 subsumes case 1. However, in general there might be situations which satisfy case 1, while not satisfying case 2. The use of NAF should be carefully considered. A safe (but restrictive) approach to generic cases containing NAF is to allow them only on the lowest level of subsumption hierarchy.

Normative assessment

Article 1: It is forbidden to overtake before a pedestrian crossing.

Article 2: Ambulances may overtake under any circumstances.

In order to establish a unique identity of a case as a whole and maintain the ability of reasoning about it, it is necessary to reify cases. First let's consider two alternative approaches to representing cases.

User-specified reification. We can link all the relevant individuals to the same instance representing the case. For this purpose we can use the axiom scheme below:

$$\textsc{Domain\_of\_Case}_i \equiv \exists \mathsf{part\_of\_case}.\{\mathsf{case}_i\} \tag{4.1}$$

Further, we specify the case description as follows:

vehicle_1:	$\textsc{Car}$, $\textsc{Domain\_of\_Case}_1$
vehicle_2:	$\textsc{Ambulance}$, $\textsc{Domain\_of\_Case}_1$
road_object_1:	$\textsc{Pedestrian\_Crossing}$, $\textsc{Domain\_of\_Case}_1$
	overtakes(vehicle_2, vehicle_1)
	before(vehicle_2, road_object_1)

Observe that each relevant individual is assigned to the class $\textsc{Domain\_of\_Case}_1$, and so, by the definition of 4.1, it is linked by property part_of_case to the respective instance case_1 representing the case as an entity. Finally, we give the formal representation of the normative rules:

$$\begin{aligned}\textsc{Art1\_Case}(\mathrm{v}) \text{ :- } & \textsc{Vehicle}(x) \wedge \textsc{Vehicle}(y) \wedge \textsc{Pedestrian\_Crossing}(z)\\ & \wedge \mathsf{overtakes}(x, y) \wedge \mathsf{before}(x, z)\\ & \wedge \mathsf{part\_of\_case}(x, v) \wedge \mathsf{part\_of\_case}(y, v) \wedge \mathsf{part\_of\_case}(z, v)\end{aligned}$$

$$\begin{aligned}\textsc{Art2\_Case}(\mathrm{v}) \text{ :- } & \textsc{Ambulance}(x) \wedge \textsc{Vehicle}(y)\\ & \wedge \mathsf{overtakes}(x, y)\\ & \wedge \mathsf{part\_of\_case}(x, v) \wedge \mathsf{part\_of\_case}(y, v)\end{aligned}$$

Skolem-like reification. SPARQL constructs, if managed by an external program, enable a more sophisticated technique, than the one presented above. Basically, we can use a function generating a unique URI on every call, which should be used as a Skolem term for representing an individual case. Thus generic case can be represented as an implication:

$$\forall_{\overline{x}} \varphi(\overline{x}) \rightarrow \exists_y \textsc{Case}_i(y)$$

meaning that for every distinct ground substitution of the sequence of variables $\overline{x} = x_1, ..., x_n$ satisfying the condition $\varphi(\overline{x})$ of the normative rule, there exists *a unique object* y which is an instance of a generic case $\textsc{Case}_i$.

Thus we can simplify the case description to the form:

vehicle_1:	$\textsc{Car}$
vehicle_2:	$\textsc{Ambulance}$
road_object_1:	$\textsc{Pedestrian\_Crossing}$
	overtakes(vehicle_2, vehicle_1)
	before(vehicle_2, road_object_1)

and generic cases into:

$$\textsc{Art1\_Case}(\textsc{new\_sko}) \text{ :- } \textsc{Vehicle}(x) \wedge \textsc{Vehicle}(y) \wedge \textsc{Pedestrian\_Crossing}(z) \wedge \mathsf{overtakes}(x,y) \wedge \mathsf{before}(x,z)$$

$$\textsc{Art2\_Case}(\textsc{new\_sko}) \text{ :- } \textsc{Ambulance}(x) \wedge \textsc{Vehicle}(y) \wedge \mathsf{overtakes}(x,y)$$

In order to link new Skolem terms with the constants substituted for the variables in the condition of the rule, the SPARQL construct expression should be augmented with the set of "trace" assertions of the form { `?x :part_of_case :y` } , where x is a variable from the condition and y is the newly introduced Skolem URI.

The difference between the two discussed representations should be clear. The second one does not require making any presumptions on which objects constitute a description of a single case and which belong to different cases. An existence of an individual case is acknowledged every time a matching between domain objects and generic cases occur. This might have a great importance in situations where more than one violation can be detected in the scenario described by the user. In such a situation, there will be more new case individuals generated, each concerning (and linked via traces) to different entities belonging to the described scenario. According to the former representation, we would all the time reason only with a single user-specified individual regardless the fact that there are different parts of the described world that get normatively qualified.

One "side effect" of the second approach is a problematic interaction between generated case individuals and generic case subsumption hierarchy. Let's assume that article 2 is more specific than 1[8], i.e.:

$$\textsc{Art2\_Case} \sqsubseteq \textsc{Art1\_Case}$$

The result of normative assessment of the previously presented scenario will yield two new individuals, say case_1 and case_2: the result of matching the case description against both norms. Hence, we obtain the following new assertions:

[8]This is not in fact true even if $\textsc{Ambulance}$ is a subclass of $\textsc{Vehicle}$. The two generic cases are incomparable w.r.t. subsumption. For the subsumption to hold the $\textsc{Art2\_Case}$ would have to be further specified with the following conjuncts $\textsc{Pedestrian\_Crossing}(z) \wedge \mathsf{overtakes}(x,y) \wedge \mathsf{before}(x,z)$

(1) case_1: ART2_CASE
(2) case_2: ART1_CASE
(3) case_1: ART1_CASE
part_of_case(vehicle_1, case_1)
part_of_case(vehicle_2, case_1)
part_of_case(road_object_1, case_1)
part_of_case(vehicle_1, case_2)
part_of_case(vehicle_2, case_2)

Assertion (1) comes from successful matching the situation against ART2_CASE, (2) — against ART1_CASE, while (3) follows from (1) and the previously generated subsumption hierarchy between generic cases. The situation should certainly be resolved, or otherwise the normative conflict will remain. The resolution should come down to relating (or equating) some of the case individuals, since clearly case_2 is a domain-subset of case_1. In some (average) situations the following strategy suffices:

> If a and b are two (case) individuals classified as instances of the same generic case, then a is a domain-subcase of b iff $\{x \mid \text{part\_of\_case}(x, a)\ \} \subseteq \{x \mid \text{part\_of\_case}(x, b)\ \}$. Iff a is a domain-subcase of b then a and b can be treated as the same individual.

Note however that the strategy will break in general.

The core of normative assessment can be captured in the LKIF-core ontology, by imposing the following axiom for each concept representing a generic case description:

$$\text{CASE}_i \sqsubseteq \text{QUALIFICATION}$$

where QUALIFICATION should be one of LKIF-core deontic concepts: ALLOWED, OBLIGED, DISALLOWED. The normative conflict resolution should be applied to these individual cases which get classified as instances of LKIF-core ALLOWEDAND-DISALLOWED. Violation (according to *lex specialis* criterion) can be asserted about those individual cases that satisfy the following query:

```
CONSTRUCT  { ?case rdf:type :VIOLATION }
WHERE      { ?case rdf:type ?GCase .
             ?GCase rdfs:subClassOf :DISALLOWED .
            OPTIONAL {?case rdf:type ?GCase2 .
             ?GCase2 rdfs:subClassOf :ALLOWED .
             ?GCase2 rdfs:subClassOf ?GCase .}
             FILTER (!bind(?GCase2)) }
```

The query classifies as Violations all instances of those generic cases that are DISALLOWED, but do not belong to any ALLOWED subclass of those generic cases.

Enhancements to individual case descriptions SPARQL can provide support for imposing additional integrity constraints (such as frameworks) and obtaining satisfying granularity level of information from the user.

The first point is straightforward. Since it is possible to express rule-like constructs in SPARQL, we can easily use them in order to go beyond the expressiveness of OWL. The definition of hasUncle in SPARQL could be put as:

```
CONSTRUCT  { ?x :hasUncle ?y }
WHERE      { ?x :hasFather ?z .
             ?z :hasBrother ?y }
```

In order to maintain decidability we have to guarantee the safeness condition. A good proposal called weak-safeness has been introduced in [Rosati, 2006][9], which basically disallows the use of unknown individuals in the consequents of rules. In the rule above one can adopt this condition, by the following SPARQL filter:

```
FILTER (!isBlank(?x) && !isBlank(?y))
```

which excludes bindings of $?x$ and $?y$ to blank nodes (i.e. possibly implicit individuals).

Furthermore, it should be possible to employ SPARQL constructs for enforcing users to provide certain crucial information required for assessment. This concerns predominantly resolving truth assignments to disjunction. Given the set of all (concept) literals $\mathbf{P} = \{P_1, ..., P_n, \neg P_{n+1}, ..., \neg P_l\}$ used in the normative rules, we can retrieve all assertions about an object a such that $Q(a)$, $Q \sqsubseteq T$ and $T \equiv S_1 \sqcup ... \sqcup P_k$ or $T \equiv S_1 \sqcup ... \sqcup \neg P_k$, where $P_k, \neg P_k \in \mathbf{P}$, but none of $P_k(a)$ or $\neg P_k(a)$ can be derived from the knowledge base[10], i.e. its truth value is unknown. In such cases we can ask the user to explicitly determine which of the disjuncts indeed hold, so that respective norms can be definitely applied or discarded.

This service will most probably require using some form of recursive SPARQL querying, since the RDF representation of OWL unions is defined recursively using rdf:first and rdf:rest predicates.

Explanation retrieval Finally, we can use flexible retrieval services offered by SPARQL in order to construct explanations. This part of reasoning essentially relies on the traceability of normative assessment inferences, and so the main point to be considered is the content of the CONSTRUCT part of SPARQL queries representing norms.

Roughly, given a case — an object representing an individual case — classified as a violation, we should start with retrieving all the objects involved in that case. Further, for each of these objects we should retrieve which predicate from the body of the norm corresponded to that individual: this part comprises a direct explanation, i.e. all the information that directly triggered the application of the rule. In order to provide also indirect information, comprising inference steps from the originally

[9]Weak-safeness is a relaxation of the well-known DL-safeness condition [Motik et al., 2004].

[10]Negation means here the complement.

asserted facts to norm predicates, we can use the subproperty/subclass hierarchy on the distance between original assertions and derived literals.

Bibliography

[Antoniou et al., 2005] Antoniou, G., Damasio, C., Grosof, B. N., Horrocks, I., Kifer, M., Maluszynski, J., and Patel-Schneider, P. (2005). Combining rules and ontologies: a survey. Deliverable I3-D3, REWERSE, IST-2004-506779.

[Boer, 2008] Boer, A. (2008). *Legal theory, Sources of Law & the Semantic Web.* PhD thesis, University of Amsterdam.

[Boer et al., 2005] Boer, A., van Engers, T., and Winkels, R. (2005). 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.

[Brachman et al., 1991] Brachman, R., McGuinness, D., Patel-Schneider, P., Resnick, L., and Borgida, A. (1991). Living with classic: When and how to use a kl-one-like language. In Sowa, J., editor, *Principles of semantic networks: explorations in the representation of knowledge*, pages 401–456. Morgan Kaufmann. http://www.cs.man.ac.uk/ franconi/dl/course/articles/brachman91living.ps.gz.

[Breuker et al., 2008a] Breuker, J., Casanovas, P., Francesconi, E., and Klein, M., editors (2008a). *Legal Ontologies and the Semantic Web: Channelling the Legal Information Flood.* IOS Press.

[Breuker et al., 2007] 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 et al., 2008b] Breuker, J., Kordelaar, P., and et al (2008b). Comparing models. Deliverable 2.5, estrella project, University of Amsterdam.

[Clancey, 1983] Clancey, W. J. (1983). The epistemology of a rule based system -a framework for explanation. *Artificial Intelligence*, 20:215–251.

[Forbus, 2008] Forbus, K. (2008). Qualitative modeling. In van Harmelen, F., Lifschitz, V., and Porter, B., editors, *Handbook of Knowledge Representation.* Elsevier.

[Gasse and Haarslev, 2008] Gasse, F. and Haarslev, V. (2008). Dlrule: A rule editor plug-in for protege. In *OWL: Experiences and Directions (OWLED).*

[Glimm, 2007] Glimm, B. (2007). *Querying Description Logic Knowledge Bases.* PhD thesis, The University of Manchester, Manchester, United Kingdom.

[Glimm et al., 2007a] Glimm, B., Horrocks, I., Lutz, C., and Sattler, U. (2007a). Conjunctive query answering in the description logic $\mathcal{SHIQ}$. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007)*.

[Glimm et al., 2007b] Glimm, B., Horrocks, I., and Sattler, U. (2007b). Conjunctive query entailment for $\mathcal{SHOQ}$. In *Proceedings of the 2007 Description Logic Workshop (DL 2007)*.

[Gordon, 2006] Gordon, T. (2006). Extending lkif. Deliverable 4.1, estrella project, Fraunhofer FOKUS.

[Grosof et al., 2003] Grosof, B. N., Horrocks, I., Volz, R., and Decker, S. (2003). Description logic programs: Combining logic programs with description logic. In *Proc. 12th Intl. Conf. on the World Wide Web (WWW-2003)*.

[Hoekstra, 2008] Hoekstra, R. (2008). *Ontology Representation: Design Patterns and Ontologies that Make Sense*. PhD thesis, University of Amsterdam.

[Hoekstra and Breuker, 2008] 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 et al., 2007] 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 et al., 2006] 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*. CEUR Workshop Proceedings 216.

[Horrocks et al., 2003] Horrocks, I., Patel-Schneider, P. F., and Harmelen, F. V. (2003). From *shiq* and RDF to OWL: The making of a web ontology language. *Journal of Wen Semantics*, pages 7–26.

[Klarman, 2008] Klarman, S. (2008). Summary of the specification of the Legal Knowledge Interchange Format deliverable 1.1. Deliverable 1.1 summary, estrella project, University of Amsterdam.

[Kremen and E.Sirin, 2008] Kremen, P. and E.Sirin (2008). PARQL-DL implementation experience. In *Proceedings of OWLED 2008*.

[Levesque and Brachman, 1985] Levesque, H. and Brachman, R. (1985). A fundamental tradeoff in knowledge representation and reasoning. In Levesque, R. B. H., editor, *Reading in Knowledge Representation*, pages 41–70. Morgan Kaufmann.

[Motik et al., 2004] Motik, B., Sattler, U., and Studer, R. (2004). Query answering for OWL-DL with rules. In *Proceedings of the 3rd International Semantic Web Conference (ISWC 2004)*, pages 549–563.

[Motik et al., 2005] 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.

[Parsia and Sirin, 2007] Parsia, B. and Sirin, E. (2007). SPARQL-DL: SPARQL query for OWL-DL. In *Proceedings of OWLED 2007*.

[Rosati, 2006] Rosati, R. (2006). DL+Log: tight integration of description logics and disjunctive datalog. In *Proceedings of the Knowledge Representation Workshop 2006*, pages 68–78.

[Schreiber et al., 1993] Schreiber, A. T., Wielinga, B. J., and Breuker, J. A., editors (1993). *KADS: A Principled Approach to Knowledge-Based System Development*, volume 11 of *Knowledge-Based Systems Book Series*. Academic Press, London. ISBN 0-12-629040-7.

[Schreiber et al., 2000] Schreiber, G., Akkermans, H., Anjewierden, A., de Hoog, R., Shadbolt, N., Van den Velde, W., and Wielinga, B. (2000). *Knowledge Engineering and Managament: The CommonKADS Methodology*. MIT Press.

[Sirin and Parsia, 2006] Sirin, E. and Parsia, B. (2006). Optimizations for answering conjunctive ABox queries. In *Proceedings of the International Description Logic Workshop*.

[Sirin and Parsia, 2007] Sirin, E. and Parsia, B. (2007). Sparql-dl: Sparql query for owl-dl. In *3rd OWL Experiences and Directions Workshop (OWLED-2007)*.

[Valente, 1995a] Valente, A. (1995a). *Legal knowledge engineering: A modelling approach*. IOS Press, Amsterdam, The Netherlands.

[Valente, 1995b] Valente, A. (1995b). *A Modelling Approach to Legal Knowledge Engineering*. PhD thesis, University of Amsterdam.

[van de Ven et al., 2008a] van de Ven, S., Breuker, J., Hoekstra, R., and Wortel, L. (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 et al., 2008b] 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)*.