Legal theory, sources of law and the semantic web

Boer, A.

DOI
10.3233/978-1-60750-003-2-i
Publication date
2009

Chapter 3

Knowledge Components

3.1. Introduction

Central to reusability in legal knowledge engineering is the notion of a knowledge component, in particular the knowledge component representing a source of law. Chapter 2 mainly focused on types of knowledge that should be differentiated in order to maximize the reusability of knowledge. For the knowledge component representing a source of law this means *not* including knowledge of generic knowledge domains, and *not* including a theory of legal reasoning that clearly aims at a specific problem solving competence and makes assumptions about generic problem domains for that purpose.

This chapter addresses the question what a knowledge component is in the context of the Semantic Web. Section 3.2 introduces the Semantic Web and its core technical standards, the uniform resource identifier and the Resource Description Framework, of which OWL DL – the knowledge representation language of interest in this book – is a component. Important to understand about the uniform resource identifier is that it does not meet the so-called *unique name assumption* often made in knowledge representation and information description logics. The *unique name assumption* will return in section 3.4.

Section 3.3 gives a classical description based on Levesque's notion of knowledge sources in [187] of what a knowledge component is and how a KBS interacts with one. Of special interest is the possibility of *revision* of a knowledge base as a partial account of defeasible reasoning.

In section 3.4 the OWL DL logic is characterized. The design requirements for OWL DL are irreconcilable with defeasible reasoning, given the open nature of the Semantic Web. The OWL DL resource, as oppose to a knowledge base, is explicitly intended to be automatically fetched from the Internet, combined with other OWL resources if they are consistent, and automatically reasoned on. For defeasible logics such operations have unpredictable consequences, and the Semantic Web standardization body (of the World Wide Web Consortium) consistently chooses against features that would make logical inference defeasible on the Semantic Web.

In law we do find rules that are defeasible, and that must be accounted for as knowledge with an ampliative function (cf. section 2.4.1). As explained in section 2.4.1 ampliative reasoning is essentially hypothesis generation, and hypotheses are falsifiable.

This of course does not mean that one cannot account for any form of ampliative inference in a purely deductive framework. As pointed out in section 2.4.1 ampliative reasoning is often characterized as a dual of deductive reasoning. A sufficiently articulate deductive theory of some knowledge domain may be sufficient to approach ampliative reasoning as model generation or satisfiability. Section 3.4.3 gives the example of solving the eight queens problem on a chess board: the rules of chess restrict the domain sufficiently to address the problem of placing 8 queens on a chess board as systematically generating the 92 fully articulate models by brute force of the proposition *eight queens on a chess board that do not attack eachother*. We can do this because we abstractly know the precise extension of the relevant chess terms.

In the absence of such articulate knowledge, and this is generally the case in law, as well as elsewhere, the ampliative knowledge we use for model expansion is not a

direct derivation of our conceptualization of the domain. In this case we use a strategy for navigating through an unknown space of possible models, and the rules describing this strategy are defeasible.

There are several possible perspectives (and of course the combinations of them) for epistemologically accounting for the meaning of such defeasible rules, for instance:

The autoepistemic perspective: The knowledge we have about a case – i.e. the propositions we have committed ourselves to – restrict the possible interpretations of a case, and these defeasible rules provide interesting hints about what other interesting knowledge is attainable[1] given what we know; In extension we can say that rules represent advice of the form *if we know [..], (we may) assume that [..]*;

The teleological perspective: The direction in which we want to expand our knowledge, either by making assumptions or asking questions, depends on the nature of the problem we are solving; In extension we can say that rules represent advice of the form *if we want to conclude that [..], assume that [..]*; and

The normality perspective: If we make assumptions, we prefer to assume that things are as normal as possible; the case is minimally abnormal.

In section 3.4.2 I will give an account of the first perspective, autoepistemic or introspective reasoning, interpreted as a form of interaction with the knowledge components of section 3.3. The second and third require too much epistemological commitment for the purposes of this work (cf. 2.6), although the realization of problem solving competence would not be possible without them. The second perspective is taken if one considers problem types and problem solving methods. The general epistemic framework of *minimality* (as Makinson calls it in [194]) and similar accounts of normality allow one to develop cognitively plausible and fully articulate logics of defeasibility that automatically and efficiently produce plausible and *minimal* interpretations of the case.

There are two reasons to account for the autoepistemic perspective in preference of the others:

1. We cannot get around the fact that what we know imposes at least some semantic constraint on the applicability of defeasible rules; and
2. The autoepistemic interpretation of interaction with OWL DL is used so widely in Semantic Web applications, apparently sometimes unknowingly, that we can take it for granted as a design commitment for KBS based on interaction with OWL DL knowledge components (cf. section 3.4.1).

It should however be clear that the discussion in section 3.4.2 aims to explain certain autoepistemic constraints on reasoning, and not a fully articulated reasoning method.

The subject of this book is representation of sources of law as knowledge components, while minimizing commitment to generic purpose theories aimed at realizing a certain problem solving competence. The theme that will be explored in chapter 4 is the relation between the law and intelligent – that is, knowledgeable – behaviour, with

[1] Without however accounting in any way of why a certain assumption is interesting.

the aim of finding some minimal ontological commitments we have to make to ground the law in a common sense understanding of the world.

Section 3.5.1 in this chapter explores the notion of source of law. Directly linking knowledge components to sources of law as a knowledge source, presupposes that the sources of law *mean* something, *without* an appeal to a specific epistemic competence. Referring back to the earlier observation on *doxastic voluntarism* (cf. section 2.4.1), it seems that they do in as far as the interpretation of the sources of law is not *completely* arbitrary. On the other hand a knowledge component that represents a source of law does not exhibit a specific epistemic competence relative to a specific problem type: only a KBS deployed in a specific setting does.

The law, and legal theory, appeal to a shared common sense understanding of the world – including the mental world. Section 3.5.2 is a quick survey of candidate generic purpose knowledge sources – knowledge about actions, about granularity, about written documents as carriers of information – that are a prerequisite for explaining the way law works. Since there are many alternative accounts of these, there are necessarily also alternative accounts of their application in law, each of them too specific to function as a conceptual coatrack for explaining the purpose of knowledge of sources of law.

The primary purpose of this chapter is to set up a shared understanding of what is intended with a knowledge component, and how defeasible rules interact with ontology. This chapter will also introduce logical syntax that will be used throughout this book.

3.2. The Semantic Web

Most of the work this thesis was performed in the context of the *Semantic Web*. The Semantic Web, and IT standardization in general, are often proposed in political-administrative circles as *potentially* the most effective non-legal solution for reducing the *administrative burden*, as pointed out in section 2.1.1 and [7].

The World Wide Web Consortium[2] is responsible for the technical open standards that make the World Wide Web possible: HTML, XML[3], HTTP, RDF, etc. The World Wide Web is mostly oriented towards information that is more or less directly presented to human beings: text, images, movies, etc. A major portion of Internet traffic however consists of data that does not fit this category: it is solely understood by computer programs. Both types of information may use for instance the XML standard, but there is a fundamental difference in meaning.

For most purposes there is no material difference between an XML structure retrievable from the Internet and the document, or movie, or image it describes. Only when one considers ephemeral issues such as how many copies of the document are around one starts noting a material difference between the familiar document and its simile on the computer. It has therefore become a convention that a XML data structure representing a document *is* a manifestation of that document, and not a description of it. For arbitrary XML data, for instance inventory records, exchanged between computer programs this convention however doesn't always work: they clearly *describe*

[2] `http://www.w3.org`
[3] `http://www.w3.org/XML`

something, for instance the inventory of a warehouse, but they clearly *are* not that thing.

The distinction between information as being understandable to humans and information understood only by computer programs eventually gave rise to a conceptual distinction between:

- The World Wide Web for human-understandable data; and the
- The Semantic Web for computer-understandable data.

World Wide Web standards have been designed to refer to an accessible world consisting of data structures that can (usually) be retrieved in some way over Internet protocols. Semantic Web standards often refer to an in principle inaccessible one consisting of things "out there" that can only be referred to by way of a symbol that functions as a substitute for the thing itself.

Of particular interest to the World Wide Web Consortium in its conception of the Semantic Web is *metadata*, often characterized as information about information, or data about data, and in particular computer-understandable data about human-understandable data. The primary use case for the Semantic Web is to make the World Wide Web better, more intelligent, by encoding computer-understandable information about documents. This is also the main subject of this book: computer-understandable information about sources of law.

Critics of the Semantic Web vision of standardizing metadata point out that the same computer-understandable information can be described in very many ways, that schemas are not viewpoint-neutral, and that people are generally stupid, lazy, and opportunistic liars (cf. for instance [98]). While most of these points have already been addressed here in some form, the point about opportunism is specific to the Semantic Web:

> Metadata exists in a competitive world. Suppliers compete to sell their goods, cranks compete to convey their crackpot theories, artists compete for audience. Attention-spans and wallets may not be zero-sum, but they're damned close. [..] Meta-utopia is a world of reliable metadata. When poisoning the well confers benefits to the poisoners, the meta-waters get awfully toxic in short order. **In [98]**

This perhaps explains why the Semantic Web so far has not made a big impact on the general web[4]. Semantic Web standards have however been adopted quite enthusiastically by the knowledge engineering community for KBS development, with the understanding that the reliability of Semantic Web data retrieved from the Internet depends on the reliability of its source: a KBS does not use arbitrary metadata available on the Internet. Instead some fixed content syndication arrangement, involving a license fee, is expected, or imported knowledge components come from a trusted standardization body.

The World Wide Web uses the same identifiers – uniform resource identifiers[5] (*URI*) – as the Semantic Web. The fundamental difference is that the URI in the World Wide Web is used to attach identifiers to XML data structures and other

[4]But note that for instance RSS (Really Simple Syndication, RDF Site Summary) is actually used extensively under the hood of many large web sites.

[5]`http://tools.ietf.org/html/rfc3986`

information objects in order for software to refer to and locate these objects, and in the Semantic web to refer to things in an often inaccessible domain of reference "out there" that are *described* by way of XML data structures (see figure 3.1), which in turn can be identified by URI and located on the World Wide Web.

The URI is obviously not a unique identifier: there is no bijective relation between the identifier and the thing it identifies.

The URI starts with a domain name, which has a *legal user* who has a leasing agreement with a domain name registrar. If users respects each other's legal rights, and legal users use each URI only once to refer to something, then URI are globally unique identifiers. This is also one of the reasons why URI cannot possibly bijective: the legal user of the URI would obtain a unique privilege relative to the object identified if no other URI could be made to identify that object.

3.2.1. The Resource Description Framework

The core Semantic Web standard is the Resource Description Framework[6] (*RDF*). Compare the following two XML statements:

1. `<name xml:base=‘‘http://foo.org/foo’’ xml:id=‘‘bar’’>Alexander</name>`
2. `<xmlNameElement rdf:about=‘‘http://foo.org/foo#bar’’>`
   `<rdf:value>Alexander</rdf:value></xmlNameElement>`

The first one is an XML element defined in some XML schema with a URI identity marker `http://foo.org/foo#bar`. The URI identifies the XML element `<name>Alexander</name>`, and not its value "Alexander" or the fact that the value is a name. The second statement is RDF, and encodes a (rather trivial) statement *about* the thing referred to by the identity marker `http://foo.org/foo#bar`, being `<name>Alexander</name>`. This difference in interpretation cannot be inferred from the XML but is a difference in the semantics of basic XML (specifically `xml:base` and `xml:id`) and RDF.

The interface between retrievable XML and the outside world consists of these URI. The document as manifested in XML *is* what the URI refers to, while RDF *describes* the thing referred to by the URI. Obviously it is possible to describe everything, including the structure, contained text blocks, references, etc. in the document format in RDF.

In this book we use URIs as constants for terms, for individuals, for variables, for information items in general, for doxastic entities *qua* information items (as discussed in section 2.4), and for ontologies and knowledge components.

In place of (absolute) URI one may use URI references: these resolve to a URI relative to a processing environment. For instance the URI reference `#bar` resolves to `http://foo.org/foo#bar` if the XML base in scope is `http://foo.org/foo`. Often the notation `foo:bar` is used, where `foo` is a *namespace*[7] declaration for URI reference `http://foo.org/foo`, and `bar` is a local name. The combination of both resolves

[6] `http://www.w3.org/RDF`

[7] http://www.w3.org/TR/REC-xml-names/

to a valid URI reference `http://foo.org/foo#bar`. The namespace mechanism is important for partitioning vocabularies into modules.

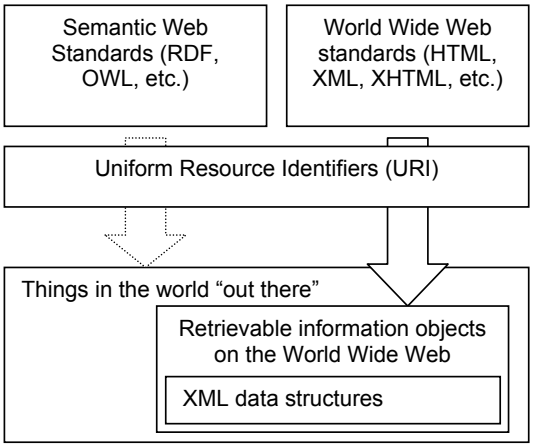

Figure 3.1. Semantic web vs. World Wide Web: one refers to an accessible world consisting of XML data structures and other information objects that can be retrieved, and the other one to an inaccessible one consisting of things "out there" that can only be referred to by way of a symbol.

An RDF description of a resource consists of a set of statements. An RDF statement is a triple with the following components:

subject: the thing the statement describes;
predicate: a property of the thing;
object: the thing the statement says is the value of the property, for the thing the statement describes.

A set of triples that share one of more URI form a labeled graph. The subject and the property value are always URIs that identify something, in the case of metadata a *resource*. The object is either a URI or a (optionally datatyped) literal. A literal identifies itself. Here I will use the notation `(predicate subject object)` for triples (or `(predicate subject "literal")` if the object is a literal). See the RDF specifications[8] for details on RDF literals and datatyping of literals.

The earlier example resolves to two triples:

1. `(foo:bar rdf:type xmlNameElement)`
2. `(foo:bar rdf:value "Alexander")`.

RDF statements may be reified and identified by a URI. The reification `r` of `(foo:bar rdf:type xmlNameElement)` is:

```
{(r rdf:subject foo:bar), (r rdf:predicate rdf:type),
(r rdf:object xmlNameElement), (r rdf:type rdf:Statement)}
```

[8]See `http://www.w3.org/2001/sw/RDFCore/`

It is however not prescribed by the standard that the following holds in the other direction:

$$\{\texttt{(r rdf:subject foo:bar), (r rdf:predicate rdf:type),}\\ \texttt{(r rdf:object xmlNameElement), (r rdf:type rdf:Statement)}\}\\ \models \texttt{(foo:bar rdf:type xmlNameElement)}$$

Reification can be used to for instance indicate the author of a statement, or to add epistemic qualifications about the statement.

An RDF aware application can correctly parse RDF and manipulates the resulting data structure in accordance with RDF semantics. Of specific interest to us is the combination of RDF and the OWL DL subset of the *Web Ontology Language*[9] (confusingly, *OWL*) that can be interpreted as a description logic. OWL DL is stored in the form of RDF data structures and processed and manipulated in accordance with RDF semantics.

OWL DL is discussed in the next section. An OWL DL reasoner (e.g. Pellet, Racer, Fact, Fact++, Kaon) is an RDF aware application that applies OWL DL semantics to RDF data structures. Wellknown editors for OWL DL are for instance Topbraid Composer, Protege, and Altova Semanticworks.

There are many features that are common in knowledge representation languages that are missing in OWL. This has led to multiple initiatives to create an additional knowledge representation language for the Semantic Web (e.g. RIF, SWRL) to be used as an alternative for, or in conjunction with, OWL. So far these either fall short of expectations or sacrifice design requirements of the Semantic Web, such as decidability and monotonic logical entailment *with* the so-called open world assumption explained in section 3.4.1.

The need for an alternative knowledge representation language is felt because knowledge engineers are aware of the existence of kinds of knowledge that cannot be properly captured with OWL DL. The introduction of this chapter qualified this knowledge as *ampliative*, and generally defeasible, in character. It is not plausible, given the existing variety in epistemological approaches to this problem, that the World Wide Web Consortium will be able to specify a logic-based language that pleases everyone.

Of special interest for the purposes of this book is introspective, or autoepistemic, reasoning, to be discussed in section 3.4.2. As will be shown in that section, introspective reasoning is already implemented using existing Semantic Web technologies, but as a method of interacting with OWL DL knowledge components, and not as a complete logical language in its own right.

3.3. Knowledge Components and Revision

The idea of reasoning as autonomous process modeled by logical entailment meshes well with the notion of a knowledge base as a black box *knowledge source*[10] interacted

[9] http://www.w3.org/TR/owl-features/

[10] This use of knowledge source conflicts with the idea of a knowledge source as the origin of the thing to be represented (i.e. a written source or an expert).

with by way of ask and tell operations (specified in detail by Levesque in [187]) using a controlled vocabulary taken from the ontology the knowledge source uses. The black box knowledge source does not itself initiate ask and tell operations. A tell operation expands the belief base with a new proposition. The OWL DL knowledge component as conceived in this book is the static knowledge base, a *belief base*, that can be loaded into a generic OWL DL reasoner, becoming a knowledge source in Levesque's terms. The black box interpretation of knowledge components in Levesque's terms will be used for at least:

1. ontology and
2. abstract "scientific" theories.

Other knowledge sources are however less well-behaved: the represented belief base is defeasible for a variety of reasons, or they resolve a query by performing ask operations of other knowledge sources whose behaviour cannot be guaranteed (for instance a human user). This makes them less freely reusable. At least the following modes of reasoning introduced in chapter 2 belong to this category:

1. abstraction, refinement, aggregation;
2. choice between a menu of actions; and
3. abductive explanation of situations and outcomes of actions.

We can conceive of these other knowledge sources in terms of their interaction with standard OWL DL knowledge sources, and explain defeasibility in terms of *belief revision* on the OWL DL knowledge source.

In linguistics and psychology it is common to think of a *proposition* as the content of a speech act, as something carried by a medium, as the content of a belief, or as something towards which we have a *propositional attitude*. The sources of law are carriers of information, and (therefore) propositions in the linguistic sense. If the relevant knowledge components carry a set of sentences S, a (rational) system applying these sentences as rules is *committed* to the set of sentences $Cn(S)$ where Cn is some operation of logical consequence.

In logic, and usually in computer science, the sentences in S and $Cn(S)$ are not explicitly distinguished: both are considered propositions of equal standing even though the ones not in S have possibly never been believed in the true sense.

In belief revision terms, the set S denotes a *belief base* K, while $Cn(S)$ is the corresponding *belief set* (cf. [134]). The beliefs in K are *occurrent beliefs*, while the other ones are *dispositional beliefs*, i.e. only held when asked about them. $K \models \phi$ if ϕ is in the belief set.

As far as ontology and abstract theory is concerned there is no appreciable difference between occurrent and dispositional beliefs. Ontologies and abstract theories can be freely composed into knowledge sources if designed properly. Given two belief bases K_1 and K_2, $K_1 \cup K_2 \models K_1$ and $K_1 \cup K_2 \models K_2$. Note that if ontology K_1 imports K_2, then $K_1 \models K_2$. An ask operation, or query ϕ to the KB is $K \models \phi$. A tell operation ϕ to K replaces K by $K \cup \phi$ if $K \cup \phi$ is consistent.

Definition 1. A *knowledge component* is a set of sentences S that denotes a consistent belief base K of a knowledge source. A *knowledge source* is committed to a consistent

belief base K; Interaction with the knowledge source takes place through ask and tell operations on K.

For defeasible rules the situation is different. As concluded in chapter 2 defeasible rules generally bridge ontological strata; A set of such rules is the interface to an ontological stratum from another stratum (or other strata). The conclusion of a defeasible rule can be held as a belief as long as it is consistent to do so. The application of a rule as a *if* ψ *then* ϕ construct through interaction with the tell and ask interface of a classical knowledge source can be conceived of as an interaction that first asks of the knowledge source whether ψ is true and if it is tells the defeasible conclusion ϕ.

A tell operation may *revise* the KB if the KB holds defeasible beliefs and the KBS prefers the told information ϕ over (some) of the beliefs in its own belief base. If $K \models \neg\phi$, K must first be modified by calculating the set of prime implicants of $\neg\phi$, and then retracting at least one belief from *each* prime implicant (cf. [134]):

Definition 2. Let $K \models \phi$ where ϕ is a sentence and K a belief base. A fragment $K' \subset K$ is a *prime implicant* of ϕ in K if $K' \models \phi$ and $K'' \not\models \phi$ for every $K'' \subset K'$. The pair (K', ϕ) could is an *argument* in favour of ϕ, relative to K.

See for instance [215] for methods to enumerate prime implicants.

This procedure is the basis for *belief base revision*, and not the more general belief-set-based belief revision approach (cf. generally [117])that dominates the literature. The decision whether and which belief to retract is based on an application specific *incision function* (cf. [134]) which implements an epistemic entrenchment theory based on meta-properties of the beliefs. If a prime implicant for instance contains only terminological axioms the KBS is committed to believe anything entailed by them, and cannot revise the knowledge base. If a member of the prime implicant is for instance a *default*, the KBS *by default* believes the entailed conclusions, and may retract the default.

Do note that I advocate making few commitments to the meta-properties of beliefs and the incision function, unless the source of law makes them. As suggested throughout chapter 2 we can distinguish between terminological axioms and axioms from abstract theories on the one hand, and defeasible rules on the other hand. This is a clear commitment. For defeasible rules it makes sense to impose a preference relation on them, if such preferences are expressed in the sources.

The belief base revision approach can provide a semantic basis for more liberal models of user interaction than the classical ask and tell approach allows, but this subject is beyond the scope of this book.

In this book, reasoning with, and importing, and consistency checking of knowledge components is assumed to be implemented in Semantic Web standards-based technology. Semantic Web standards have been designed with this goal in mind, and, in as far as they are successful as a standard, other non-legal knowledge components, that play a supporting role in legal reasoning, will also be available in this form. Section 3.4 discusses the OWL description logic, and the idea behind it.

Section 3.4.2 discusses an approach to representing knowledge that is subject to revision, and interpreting its meaning as a form of autoepistemic, or introspective,

reasoning. This interpretation is based in the observation that there is something peculiar about the way in which the *ask* operation is often implemented in Semantic Web applications. It is a characterization and justification of an already existing approach to using Semantic Web technology in a KBS rather than a proposal of one. In this book defeasible rules are only interpreted in relation to the (non-defeasible) knowledge component to which they belong:

Definition 3. A *defeasible extension to a knowledge component* is a set of defeasible rules, interpreted as recommended or sanctioned but defeasible ask/tell interactions with a knowledge source committed to the belief base represented by that knowledge component.

3.4. Description Logic, OWL DL, and Autoepistemic Interaction

Let us start with characterizing some identifying features of *description logic*, the language used for describing legal knowledge in as far as that is possible. Of interest here is the specific description logic OWL DL.

A DL *knowledge base* consists of a *tbox* – a set of terminological axioms – and a *abox* – a set of grounded axioms about a particular world of named individuals called *assertions*.

The tbox offers a framework for subsumption and classification, and the abox contains statements about particular things or *individuals* in the world. These *things* are denoted by constants. The distinction was introduced by Levesque in [187], in an excellent overview of the functions of knowledge bases, but has gradually changed in meaning somewhat.

The DL knowledge base is intended to be used as a knowledge source as specified in the previous section. It is however common to restrict the possibility of *telling* propositions to the knowledge base to assertions in the abox in normal interaction scenarios.

The description language is composed of:

1. constants denoting terms;
2. constants denoting properties;
3. constants denoting individuals;
4. assertions applying terms to individuals and properties to pairs of individuals; and
5. terminological axioms constraining the application of terms to individuals and properties to pairs of individuals.

Each term denotes a set of individuals $\{i_1, \ldots, i_k\}$, each property a set of pairs $\{(i_1, i_2), \ldots, (i_j, i_k)\}$ of individuals.

In this book I use a typographic distinction between CONCEPTS, properties, and generic constants or variables a, b if ambiguity could arise. Alternatively `concepts`, `properties`, `a`, `b` may be used. If I need namespace (ns) qualification, the notation `ns:c` is used. The following is a typographical example of a terminological axiom:

$$\textsc{Function} \equiv \exists \mathsf{playedBy}.\textsc{PhysicalObject} \sqcap \neg \textsc{Agent}$$

Something is a function if and only if it is played by some physical object, and not an agent. The following is an assertion:

FUNCTION(`example:c`)

The individual denoted by c in namespace ns is a function. Description logics vary mostly by expressiveness. Differences can therefore often be characterized in terms of syntax only. There are however some notable differences between OWL DL and most other description logics that can be explained in terms of a few assumptions underlying the logic, which will be addressed later.

Some readers will be familiar with the modal logic-based semantics of most description logics. To facilitate the understanding of DL axioms for those that are not, I present a translation to FOL based on [35]. See [11, 13, 12] for an overview of efficient tableau methods for description logics, including OWL DL.

OWL DL is an expressive description logic recommended for use as an XML standard by the World Wide Web Consortium[11]. A new proposal, OWL 2[12], extends the expressiveness even further, but this extension will not be systematically taken into account in this book as it was very new when this book was completed. The description logic embodied by OWL DL can be properly understood as a labeled modal logic, to be characterized by modal frames (cf. [157, 158]).

The set of description logic classes $\mathcal{DL}$-*CLASS* covered by OWL DL is defined by the following rule, where C, C_1, C_2 are classes, P, P_1, P_2 are properties, i_1, i_k are abox constants, and N is a name:

$$\begin{aligned}\mathcal{DL}\text{-}Class ::= & N|\top|\bot|\neg C|C_1 \sqcap C_2|C_1 \sqcup C_2|\forall P.C|\exists P.C|\{i_1, \ldots, i_k\} \\ & |n \leq P|n \geq P|n \leq P.C|n \geq P.C\end{aligned}$$

The DL class corresponds closely with the notion of *concept* in section 2.3. The construct $\{i_1, \ldots, i_k\}$ is a *nominal*, a concept that denotes the set of individuals denoted by the constants in the given set[13] (see [253] for a detailed discussion of nominals).

Cardinality restrictions (the last four) will not be used in this book. See table 3.1 for an explanation of the syntactic subset used.

In addition, there is a set of properties $\mathcal{DL}$-*PROP* (see table 3.2) used in property axioms that allow us to specify the nature of the intended relation, by restricting domain and range, and specifying i.a. functional, symmetric, and transitive frames. These are not completely FOL expressible. They are also of less interest because property axioms are not used in this book. There are important syntactical restrictions on which frame properties can be *combined* for the same property, and how complex properties may be used in class axioms.

There are class axioms and property axioms. The set of tbox axioms $\mathcal{DL}$-*TBOX* for $\mathcal{DL}$ (see table 3.3) is defined by the following rule:

$$\mathcal{DL}\text{-}TBOX ::= C_1 \sqsubseteq C_2|C_1 \equiv C_2|P_1 \sqsubseteq P_2|P_1 \equiv P_2$$

[11]Described in `http://www.w3.org/TR/owl-features/`

[12]`http://www.w3.org/TR/owl2-primer/`

[13]I.e. every instance of this concept can be denoted by one of the members of the set.

where C_1, C_2 are classes in $\mathcal{DL}$-$CLASS$, and P_1, P_2 any property.

The set of abox axioms, or *assertions*, $\mathcal{DL}$-$ABOX$ for $\mathcal{DL}$ (see table 3.4) is defined by the following rule:

$$\mathcal{DL}\text{-}ABOX ::= C(a)|P(a,b)|a = b|a \neq b$$

where C is a class in $\mathcal{DL}$-$CLASS$, a, b are abox constants, and P a property in $\mathcal{DL}$-$PROP$.

The notion of a DL model is defined as usual on the basis of multimodal Kripke models. $M_{\mathcal{DL}} \models \varphi$ means that a formula φ is true in DL model $M_{\mathcal{DL}}$. The truth definition for this language, and semantic notions such frame, satisfiability and validity are standard (see [24]), subsuming propositional logic (the $\sqcup$ and $\sqcap$ operators are only syntactically different from $\vee$ and $\wedge$), extended with accessibility of individuals which function as the *worlds* of standard modal logic definitions:

$$M_{\mathcal{DL}}, i \models \exists P.\phi \text{ iff } \exists i' : P(i, i') \text{ and } M_{\mathcal{DL}}, i' \models \phi$$

The notation $\forall P.\phi$ is an alternative for $\neg\exists P.\neg\phi$. Term C applies to individual i is written $C(i)$ as an assertion, which is true if $M_{\mathcal{DL}}, i \models C$.

OWL syntax is based on RDF triples. $C(x)$ for instance translates to the triple `(x rdf:type C)`, and $C_1 \sqsubseteq C_2$ to `(`C_1 `rdfs:subClassOf` C_2`)`. The standard description logic syntax used in this book can be automatically generated from OWL using the OWL Latex Tool from `http://owltools.ontoware.org/`. The mapping of description logic syntax to an OWL graph is not always immediately evident, since certain evident notational conventions used in DL notation such as operator precedence and the use of parentheses don't work in OWL's triple-based syntax.

Three important assumptions must be known to understand OWL DL reasoning.

Firstly, the sets of constants denoting terms, properties and individuals are assumed to be disjoint in OWL DL: let us remember this as the *disjointness of universals and particulars assumption*. Terms have instances, individuals do not. The disjointness assumption is a typical condition for the realization of epistemic competence: without it the logic is not decidable. Strictly speaking, this assumption even prohibits us from for instance annotating terms with a human-readable label. The disjointness of universals and particulars assumption prohibits reification of propositions or terms as individuals.

There is good reason to occasionally violate against it, and there are simple solutions for carving up a non-DL problem into DL subproblems (cf. generally [126]). The disjointness of universals and particulars assumption is normal for the description logic world, but RDF does not have it. One doesn't usually combine OWL DL with reified RDF triples. For epistemic-level annotation purposes (recommended for defeasible reasoning, as pointed out in section 3.3) one would however want to do so. Combining OWL DL semantics with annotation of reified triples is in principle not a problem. Note that the proposal to extend OWL to OWL 2 introduces both "entity annotations" and "axiom annotations"[14].

[14]See `http://www.w3.org/TR/owl2-primer/`.

Secondly, contrary to many other description logics, the presence of two different constants in the abox does not denote the presence of two different things (i.e. the *unique names assumption* does *not* hold): in OWL we must state specifically that $a = b$ or $a \neq b$. A constant denotes one unique thing, but a thing may be denoted by multiple constants. This means that a set of constants $\{i_1, \ldots, i_k\}$ denoting individuals does not in itself give an indication about the set of individuals it denotes: we have to state explicitly that all constants denote different individuals.

Thirdly, OWL DL is based on the *open world assumption*. As will be explained in the next section, where the difference between open and closed world will be addressed, the set of assertions in the abox is often intuitively interpreted as a description of a specific Kripke model of the tbox. Because of the (no) unique names assumption and the open world assumption, this is not generally the case in OWL DL. If K_{DL} is the belief base denoted by a set of OWL DL sentences, $K_{DL} \models \neg\phi$ and $K_{DL} \not\models \phi$ are not generally equivalent, as would be the case under the closed world assumption.

$\mathcal{DL}$-*Class*-Syntax	$\mathcal{FOL}$-Syntax
$\top$	$C(x) \vee \neg C(x)$
$\bot$	$C(x) \wedge \neg C(x)$
C	$C(x)$
$\neg C$	$\neg C(x)$
$C_1 \sqcap C_2$	$C_1(x) \wedge C_2(x)$
$C_1 \sqcup C_2$	$C_1(x) \vee C_2(x)$
$\forall P.C$	$\forall y(P(x,y) \rightarrow C(y))$
$\exists P.C$	$\exists y(P(x,y) \wedge C(y))$

Table 3.1. Correspondence between $\mathcal{FOL}$ and $\mathcal{DL}$-*Class* syntax

$\mathcal{DL}$-*Property*-Syntax	$\mathcal{FOL}$-Syntax
P	$P(x,y)$
$\neg P$	$\neg P(x,y)$
$P_1 \sqcap P_2$	$P_1(x,y) \wedge P_2(x,y)$
$P_1 \sqcup P_2$	$P_1(x,y) \vee P_2(x,y)$
$P_1 \circ P_2$	$\exists z(P_1(x,z) \wedge P_2(z,y))$
P^-	$P^-(y,x)$
$P \upharpoonright C$	$P(x,y) \wedge C(x)$
$P \downharpoonright C$	$P(x,y) \wedge C(y)$
P^+ and P^*	non first-order expressible

Table 3.2. Correspondence between $\mathcal{FOL}$ and $\mathcal{DL}$ property syntax

Although OWL is quite expressive for a description logic, it does have clear limitations that many other logic-based languages do not have. An OWL DL terminological axiom can only enforce a treelike structure of individuals, since there is no provision for variable binding or for local reflexivity (the Self concept; cf. for instance [180]).

This obviously also means it is only possible to specify axioms over pairs of constants that are in the extension of some property. A sentence such as $\forall x \exists y \mathrm{C}(x) \rightarrow \mathrm{D}(y)$ specifying constraints over things that may be unconnected to each other is

$\mathcal{DL}$-*TBOX*-Syntax	$\mathcal{FOL}$-Syntax
$C_1 \sqsubseteq C_2$	$\forall x C_1(x) \rightarrow C_2(x)$
$C_1 \equiv C_2$	$\forall x C_1(x) \leftrightarrow C_2(x)$
$P_1 \sqsubseteq P_2$	$\forall xy P_1(x,y) \rightarrow P_2(x,y)$
$P_1 \equiv P_2$	$\forall xy P_1(x,y) \leftrightarrow P_2(x,y)$

Table 3.3. Correspondence between $\mathcal{FOL}$ and $\mathcal{DL}$ tbox axiom syntax

$\mathcal{DL}$-*ABOX*-Syntax	$\mathcal{FOL}$-Syntax
$C(a)$	$C(a)$
$P(a,b)$	$P(a,b)$

Table 3.4. Correspondence between $\mathcal{FOL}$ and $\mathcal{DL}$ abox axiom (assertion) syntax

semantically suspect, and undesirable. This restriction on expressiveness is not one we should try to solve: a *case* to be solved should be characterized by a completely connected graph in the abox, and unconnected entities cannot "influence" each other. If one nevertheless wants to use such axioms, one relates all constants to each other with a generic related property: $C \sqsubseteq \exists \mathsf{related}.D$.

The most obvious and most often proposed extensions to expressiveness are directed towards:

modeling more general relational structures: this is usually conceptualized as a freer use of variable binding. This comes down to being allowed to say that two nodes in the tree are the same thing in our domain, i.e. expressions along the lines of $\forall x \mathrm{C}(x) \rightarrow \exists y(\mathsf{self}(x,y) \wedge (x = y))$.

introducing ternary or n-ary properties: this is intended to represent tuples with more than three places. This can also be achieved by way of reification of the property as a class, for instance the 4-place tuple give(AGENT, RECIPIENT, THEME) to a class axiom GIVING $\sqsubseteq$ ($\exists$agent.$\top \sqcap \exists$recipient.$\top \sqcap \exists$theme.$\top$). This however leads to additional requirements on expressiveness of constraints on relational structures, the previous item on this list, since certain features that can be expressed succinctly about properties – for instance symmetry – cannot be expressed in an equivalent form on properties reified as a class.

More general relational structures can be represented with various existing rule languages that can be used with OWL DL. The at the moment of writing new OWL 2 proposal also radically expands the relational structures that can be represented in axioms through the use of role inclusion and local reflexivity (Self) (cf. for instance [180]), provided that the abox is preprocessed to relate all known constants with a generic top level property such as related, and adds axioms of the form $C \equiv \exists \mathsf{c}\textsc{Self}$.

It is generally accepted that n-ary properties are not strictly required for expressive purposes. Reifying n-ary properties, in accordance with the W3C Working Group Note of 12 April 2006 on *Defining N-ary Relations on the Semantic Web*[15], is a preferred solution.

[15] http://www.w3.org/TR/swbp-n-aryRelations/

Another apparently desirable extension is an extension on the epistemic level, which approaches reasoning on OWL DL knowledge bases with a different set of assumptions than the one outlined above. That the need for this apparently exists is clear if one looks at the methods used for interacting with OWL DL knowledge bases. OWL DL conformant description classifiers usually consist of a description logic theorem prover and a – usually more expressive or in any case different – *query language* like SPARQL (cf. generally [159]).

Abstractly a query language based on OWL DL's satisfiability test can determine whether some (closed) proposition φ is or is not entailed by some DL belief base K_{DL} ($K_{DL} \models \varphi$ and $K_{DL} \not\models \varphi$; cf. generally [159, 157]). More interesting for software development is the derived possibility of determining for which constants i, i' in the DL abox some open proposition $C(i)$ or $P(i, i')$ is true ($K_{DL} \models C(i)$ and $K_{DL} \models P(i, i')$). These open queries, for instance SPARQL SELECT queries, return a result set. These i, i' in the result set are the *known constants* in the abox denoting individuals, to be interpreted with the open world assumption and no unique name assumption in mind, not a complete set of distinct individuals.

If one however *misinterprets* the meaning of the result set in certain ways, one may obtain interesting new results. Facilities like SPARQL SELECT can for instance be used to interface a Datalog-style reasoner to an OWL DL knowledge source in the background. A common *hybrid* design in expert systems is to use a terminological component, an OWL DL knowledge source, in the background, and a datalog-based production rule component in the foreground that queries and updates the terminological component. Other such hybrid architectures replace Datalog with a default logic reasoner or a constraint checker querying and updating OWL DL knowledge sources.

The interpretation these hybrid architectures give to the abox of the OWL DL knowledge source is usually autoepistemic or, with another word, introspective. This observation (taken from [205]) – and a minimal set of constraints following from it – is the basis for the interpretation of defeasible rules in this book.

3.4.1. Autoepistemic Querying and Datalog

While the knowledge a reasoner has about the world is necessarily incomplete, it is reasonable to assume that the reasoner always has complete and accurate knowledge about what it *knows* (cf. for instance the *assumption of closure* in [187]).

When a logic acknowledges that the content of our knowledge base is only a subset of the relevant information that *could* be available about the things in the world that have our interest, we speak of *open world reasoning*. Knowledge representation languages are often characterized by open world reasoning. Since this conception of reasoning is also more suitable to the open ended Semantic Web, OWL DL is also limited to open world reasoning.

The basic idea of traditional logics, including OWL DL, is that the domain being modeled can be abstractly conceived of as a *model*, a set of objects and relationships between them. There are many possible models of the domain: a logical theory, in the case of OWL DL usually a tbox and an abox, admits reasonable models and rules out unreasonable ones. The OWL DL belief base is not a complete representation of the

domain being modeled.

Information and models are separated in this paradigm. The meaning of information is in the mapping between the information and the set of models that are consistent with it. The information tells us that *the* model is in this set. Entailment in this paradigm comes down to checking if a proposed piece of information holds in *all* models that are consistent with the available information. Adding information further restricts the set of possible models.

Many KBS however use another paradigm: that of the database. The relational model underlying databases also models the domain in terms of objects and relationships between them, but makes some radically simplifying assumptions: the only relevant objects and relationships that exist in the domain are those captured by the database information. This is the *closed world assumption*. Secondly, the names in the database uniquely identify objects in the domain, i.e. no two constants refer to the same object. This is the *unique name assumption*.

The result of these assumptions is that there is just *one* model of the domain, the one captured by the available information. There is no distinction between the model and the information about it. This is obviously very convenient, but not always very realistic.

Datalog is a formalization of the database approach in which Horn rules (viz. for instance [220, 205]) are used to capture both the structural constraints on models of the domain and the model itself. The objects and relationships in the model are only those explicitly mentioned in rules.

Rules with empty bodies are often called *ground facts*. A rule with an empty head is often called a *constraint*, and is used to express the fact that a model satisfying the condition described in the body of the rule is not admitted. Usually rules must be safe: a *safe rule* is one where all of the variables in the head of a rule also occur in the body of the rule.

Plain datalog does not include negation, but it is very often extended with a similar construct: *negation as failure* in the body of rules. In keeping with the closed world assumption it is used to draw conclusions about what is not true from what is not in the database, i.e. not known. In this book the notation **fails** is used for negation as failure instead of the classical $\neg$. Datalog is often used in conjunction with OWL DL exactly because OWL DL cannot draw conclusions from failure.

Restriction of a set of possible models by adding information is not possible in datalog (cf. [220]). Adding non-constraint rules does not restrict the set of possible models, but instead *changes* the model. Constraint rules can be used to make certain *changes* impossible.

An OWL theory describes a *set* of models M, and a datalog program describes a specific model m. If one proposes to interrogate an OWL theory *as if* it were a set of datalog ground facts, which is what often happens in reality if a Datalog-style reasoner uses an OWL DL knowledge source as a quasi-database, one might naively expect to obtain some model $m \in M$, but this is not the case.

Example 2. Take for instance an OWL theory consisting of abox $C(a)$ and tbox $C \sqsubseteq \exists \mathsf{p}\top$; In the understanding that x, y are universally quantified variables, a query $C(x)$ binds x to $\{a\}$, and a datalog query $p(x, y)$ produces an empty result set $\{\}$.

This model is however not in M. Or take an OWL theory consisting of abox $P(a)$ and tbox $P \sqsubseteq Q \sqcup R$. Models of this theory would classify the object referred to by a as at least a P, Q or a P, R. An expanded OWL theory consisting of abox $P(a)$ and tbox $P \sqsubseteq Q \sqcup R, Q \sqcup R \sqsubseteq S$ would entail $S(a)$, but a datalog program $S(x) \leftarrow Q(x), S(x) \leftarrow R(x)$ would not, as similar as the Datalog program may seem to $Q \sqcup R \sqsubseteq S$.

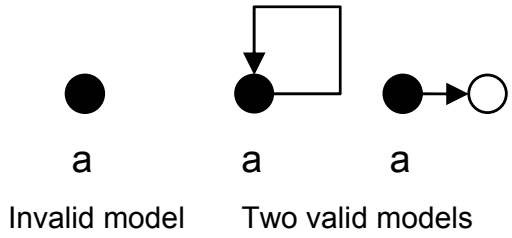

Figure 3.2. Models of C. Solid arrows represent property p. Datalog queries interrogate a model that is not a valid model of C. Instead it represents the settled part of *all* models of C.

The model m interrogated by the Datalog reasoner only consists of the objects and relationships we *know*, i.e. have explicitly represented as a constant in our theory about the domain. It is of course possible to use a set of Datalog rules to update an OWL DL knowledge source with tell operations as if it were a database, but only if the expansions one makes are within the space of possible models. If the Datalog program is carefully handcrafted this may be done successfully, but the maintenance of this Datalog program will be a non-trivial matter.

This mode of interaction between Datalog and OWL DL can be characterized as autoepistemic or introspective reasoning: it gives information about the part of *the* model that is settled, the objects and relationships that are in *all* possible models of the theory. This mode of inference is useful as an ingredient for *ampliative reasoning*, the selective expansion of the part of the model that is settled until we have solved a problem (confer with section 2.4.1 on ampliative reasoning). The expansions of the model we are interested in depend on the problem setting: outside of this problem setting this knowledge is not reusable. Which expansions we prefer largely depends on what is in the settled part of the model, i.e. on the available knowledge.

Note that this autoepistemic characterization only applies to a specific, simple approach to *interfacing* other forms of reasoning to OWL DL knowledge sources *as if* they are databases that return result sets.

This use of Datalog in conjunction with OWL DL is rather crude and inherently unsafe, and may cause inconsistency. More advanced proposals however exist that explicitly take into account defeasibility.

3.4.2. Rules, Defaults, Constraints

A pointed out in [205] there are three familiar uses of closed world reasoning that KBS developers always want to bring in to open world assumption-based knowledge representation languages:

constraints: reasoning about what is missing from the knowledge base to either drive user interaction or to check knowledge availability assumptions; for instance: *Every taxpayer has an SSN number, and John is a taxpayer, but John's SSN number is not known: therefore ask John for his SSN number*;

negation as failure: reasoning about what is *not* known about the domain to draw conclusions about what is not the case; for instance: *there exists no social security record of John, therefore, assuming all social security records are known, John is not a taxpayer*;

defaults: reasoning about what is plausible to assume to be the case, given that our database contains no information in either direction, and when to retract the assumption if the gaps are filled in later. Default rules often have the format[16] *if we know α, and it is consistent to assume β, then conclude γ*, for instance *if John has stated no income to the tax administration, and it is consistent to assume John is unemployed, then assume John is unemployed.*

Constraints and negation as failure where already mentioned in relation to Datalog interfaced with an OWL DL knowledge source as database. Even *default rules* can be used in this hybrid Datalog-OWL setup if the subject of the negation as failure operator can be a negated proposition (cf. [127]). Since OWL DL can represent equivalence and negation, direct interrogation of an OWL DL knowledge base *in effect makes this possible even if the Datalog reasoner has no explicit provision for it.*

KBS using OWL DL knowledge sources often employ some Datalog-based implementation of rules and constraints, including negation as failure, to precisely specify the intended model and to guide information collection, which generally consists of asking for the missing ground facts. In a *carefully handcrafted set* of rules this may work; Multiple such sets, with a control structure guiding their application, may be used as the implementation of a problem solving method. Datalog however has no solution for the existence of multiple, alternative expansions of a model, and the potential interactions between different directions of expansion.

Specialized logics for law often employ some mechanism for *default reasoning*, which does explicitly account for the existence of multiple, alternative expansions, and the potential interactions between default rules. Characterizations of default reasoning are generally based on the notion of *minimal* models. Minimality – when applied to sets of propositions instead of models – was already used as a criterium in section 2.4.1 for *explanation* and in section 3.3 for *argument*. As noted in section 2.4.1, the best explanation of something is often characterized as the disjunction of all *minimal* explanations, ordered by epistemic preference.

Ampliative reasoning is necessary for non-trivial problem solving as section 2.4.1 pointed out. This is however not in itself a justification for any specific approach

[16] Following Reiter's default logic (cf. generally [175]).

to representing ampliative knowledge. There is little reason to believe that good explanations are, or should be, generated by these types of rules.

In law we however have to deal with the additional problem that the relevant legal theory, and the legislation to be represented as knowledge, may contain rules explicitly dealing with reasoning on incomplete information. Reasoning with incomplete information is also a key component of dialectical argumentation: without some way to represent competing expanded models of the same settled model there is no way to even conceptualize dialectical settings.

The objective for the purposes of this book is to accommodate some representation of this type of rules in the sources of law that does not permit indiscriminate mixing closed world and open world reasoning.

One can conceive of these rule as plan operators stating advice about what is useful to *add* (without committing to the idea that this is a form of entailment) to the database given a certain objective. What something *is* on the ontological level, cannot – generally speaking – be conditional on how much we know about it: This is a matter of ontological commitment. If a conclusion is based on introspection, it has a different epistemic status than conclusions derived from terminological inference. The epistemic status of the *belief* is a property of the reified RDF statement, and of no interest in this book.

Common extensions of DLs with the function of expressing constraints and defaults can be modeled modal autoepistemic logics (cf. [205]). Autoepistemic logics model for *introspection*, the ability to reason about one's own beliefs. In these proposals, DLs are extended with an autoepistemic knowledge operator K for known and A for assumption.

The result set returned for previously discussed queries testing $K_{DL} \models C(i)$ or $K_{DL} \models P(i, i')$ for all i, i' used in the abox *is* a correct set of *known* individuals with the requested features, although the sets may turn out to contain duplicates; These queries potentially form the basis for a correct implementation of autoepistemic $K\mathrm{C}$ and $K\exists\mathsf{p}.\top$ concepts (cf. [72]), and can be explained as a kind of poor man's autoepistemic logic.

[127] point out that negation as failure $\mathsf{fails}\ C(i)$ applied to abox constant i is correctly implemented by establishing $K_{DL} \not\models C(i)$ through a (failed) query. Concept $\mathsf{fails}\ C$ has as its extension the complement of $K\mathrm{C}$ in the set of constants used in the abox. Moreover, there is no distinction between the K and A operator when applied to a set of abox constants used in a (non-epistemic) OWL DL belief base (cf. [72, 127]): concept $A\mathrm{C}$ maps to concept $\neg\mathsf{fails}\ C$, which means that any individual assumed to be in C belongs to the complement of the extension of $\mathsf{fails}\ C$.

As pointed out in [127], Reiter's default rules *if we know α, and it is consistent to assume β, then conclude γ* translate to $\mathsf{K}\alpha \sqcap \neg\mathsf{A}\neg\beta \sqsubseteq \mathsf{K}\gamma$ from an autoepistemic point of view. The complicated $\neg\mathsf{A}\neg\beta$ concept for "it is consistent to assume" evaluates to $\mathsf{fails}\ \neg\beta$.

For the representation of the conditions of standard datalog rules and constraints, negation as failure conditions, and "it is consistent to assume" conditions in default rules we can tentatively operationalize three operators known, fails, and free in terms of queries on an OWL DL knowledge source committed to some belief base.

Definition 4. For a constant i used in the abox of an OWL DL belief base K_{DL}:

1. (known.C) is true if and only if a query $K_{DL} \models C(i)$ evaluates to true;
2. (fails.C) if and only if not (known.C); and
3. (free.C) if and only if (fails.$\neg C$).

Interpreting these operators as second-order properties, we may define in OWL syntactic structures outside the DL fragment for the rule in general, and the constraint and default specifically, as meta-concepts RULE, DEFAULT, CONSTRAINT. A simple example is the prototypical default rule:

$$(\text{DEFAULT } r \ (\mathsf{known}.C_1)(\mathsf{free}.C_2)(\mathsf{assume}.C_2))$$

where DEFAULT is the rule class, *assume* is the action of telling the OWL DL knowledge source, and the rule is implicitly universally quantified over the individuals in the abox of that OWL DL knowledge source. Since we are querying on concrete sets of abox constants and pairs of abox constants, the operators could also be used with explicit universally quantified variables ((known.$C(i)$) and (known.$P(i, i')$)) without significant additional problems.

The term DEFAULT, of which r – which violates the disjointness of universals and particulars assumption – is an instance, is defined as followed:

$$\text{DEFAULT} \equiv \text{RULE} \sqcap \exists \mathsf{free}.\text{OWL:CLASS} \sqcap$$
$$\exists \mathsf{known}.\text{OWL:CLASS} \sqcap \exists \mathsf{assume}.\text{OWL:CLASS}$$

The constraint has no *assume* participant.

The rule, which is itself an object, would normally speaking be part of a larger structure, at least a set, specific to a problem solving task. The rules should be conceived of as database update plan operators, to be composed into structures specific for problem solving tasks. If these sets are purposefully composed for a specific role in an application, their application to an OWL DL knowledge source may be known to be safe – i.e. there is no risk of inconsistency.

Inserting back the result of application of an autoepistemic rule into the knowledge base, with information about its epistemic status, can be considered an intentional operation, based on explicit epistemic decision making. The appropriate tactics for this activity on general sets of rules are beyond the scope of this book. Section 3.1 laid out the technical basics of an OWL DL knowledge source revision strategy, based on the calculation of prime implicants. The reason to mention this *belief base* revision approach explicitly, is that this prime implicant calculation feature is already present and is also used for debugging OWL DL inconsistencies.

The implementations and ideas in [205, 72, 134, 168] are compatible with the approach presented here, and extend it in various directions. [175] has implemented Reiter's default logic on top of OWL DL. If we have to explicitly deal with priority of rules, the approach of [168] is worth mentioning. More advanced solutions allow the assertion of new individuals (existential introduction, or skolemization semantics for existential variables), and more generally allow for a freer use of variables.

Note that there is a great variety of semantics definitions of defeasible logic, some of which can presumably be reduced to selection of an incision function (cf. [134] for the concept of incision function) as pointed out in section 3.1, like the one in [175]. Logics in the same tradition as Reiter's are the most obvious candidates, and certainly the ones that find application in the field of computer science & law, like [211, 228], are of interest. There are however also a lot of frameworks for defeasible reasoning that are for certain not easily interpreted in terms of database updates and incision functions. See for instance [194] for a systematic inventory of a number of types of semantic characterization of defeasible reasoning, many of which go beyond the idea of theory revision.

3.4.3. Abstract, Hypothetical Objects

Neither OWL DL nor the rules introduced in the previous section allow the introduction of a new object to the *settled model*, as described in the previous section, at least not without accepting that variables occurring in the head of a rule occur in the body only in free and fails conditions. The set of constants in the abox remains the same. Constraints can only be used to *test* whether the required objects have been introduced to consider a problem solved. To introduce objects we typically have to resort to using OWL DL nominal concepts (e.g. AbstractChessPosition $\equiv \{a1, a2, \ldots, h8\}$, cf. generally [253] for discussion of the use of nominals in OWL), or we ask the user to supply a new constant with the right features.

One of the reasons for stressing the importance of the distinction between abstract things and the real things they are an abstraction of, is that the extension of abstract concepts is usually clearly delineated (although not necessarily finite). We have no idea how many chess boards and how many chess squares exist in our domain, but on the abstract level every chess position can be explained in terms of the same 1 board and 64 chess squares that we *do know*, and a problem like placing 8 chess queens on a board such that they do not attack each other becomes simply a matter of systematically trying all of them on *the* abstract board. Because the chess KBS is applying an abstract theory that explicitly spells out the relevant abstract objects in the domain, the relevant objects exist in the settled model from the outset. Any *hypothetical object* is necessarily an abstract object.

The automatic introduction of abstract objects through a strategy decided on in advance is important for planning and design systems: in such systems one does not generally ask the users to introduce the candidates, but instead explores the abstract space of abstract candidates implicit in the abstract theories being applied. In many LKBS this is not an issue, since the user supplies the "facts of the case", and the LKBS limits itself to inferring interesting (terminological and default) propositions about them.

Note that because all hypothetical objects are abstract objects, we can in principle represent the assumptions and closed world reasoning conclusions of any number of agents in the same discourse context, as long as we don't assert that the things they denote by different constants are the same thing. We can also easily test whether sets of agents have consistent beliefs by trying to equate different abstractions of the same

real world referent. We also don't have to do any bookkeeping outside of the OWL DL model to differentiate belief sets of different agents, since we can add a direct relationship between an agent and the hypothetical objects in his mind.

3.5. About Relevant Knowledge Sources

The notion of sources of law is introduced in 3.5.1, so that the reader will move to chapter 4, which is about law and its function in a social system, knowing what a source of law as intended here is.

These representations of sources of law are used in combination with more general ontologies and theories. In many cases these are perfectly generic: they have nothing to do with law as such. Subjects such as time, wholes and parts, action, do not belong to the field of legal knowledge engineering, but are frequently used in it. Section 3.5.2 reviews some existing ontologies and abstract theories.

We need to distinguish the ontological part of a theory and the defeasible part: most generic theories first require an abstraction step into the ontology, and the theory only yields useful predictions if certain knowledge availability assumptions (usually modeled with autoepistemic constraints) are met.

Section 3.5.3 for instance explains event calculus briefly. Event calculus is both an ontology, a conceptualization of what change is, and an abstract theory for filling in the blanks in an incomplete scenario, based on frame assumptions: if we do not *know* that something changed, event calculus predicts future states based on the assumption that it didn't. This is true for many such generic theories: they consist of an ontological and and autoepistemic component that is more often than not not separated. A theory like event calculus was in the first place developed to explain the use of frame assumptions in planning, not to be a conceptualization of action.

One does not ideally "extend" an abstract theory with legal concepts found in a source of law by importing the abstract theory into the representation of the source of law. The generic theory has a function, it makes a certain epistemic competence possible. The ontological purist distinguishes the following:

1. the concept C_l in the source of law;
2. the abstract concept C_a in the abstract theory; and
3. the autoepistemic abstraction of the concept C_l to C_a, or alternatively, a pragmatic $C \equiv C_l \sqcap C_a$, with the risk that C_l and C_a will turn out to become inconsistent with eachother at some point in the future because of changes made to them.

One may however generalize the concepts to which an abstract theory like event calculus can be applied into a *core ontology*. Event calculus is for instance a perfectly viable candidate component of a core ontology, if one forgets about the frame assumptions and their applications. This approach is taken in the LKIF ontology (cf. [153]).

Of importance to chapter 4 are in particular action and intention, and institutions.

3.5.1. Sources of Law

Legal knowledge acquisition only rarely directly appeals to skill, or "epistemic competence in action" as a source of knowledge to be encoded in a knowledge representation. Instead it nearly always directly appeals to written information. Why this is the case is explained in section 4.2, and how the information should be managed and interpreted is explained in section 5.3. Central to legal knowledge representation is the relation between two types of information items: arguments and the legal writings backing them. The knowledge representation represents the meaning of legal writings, and it functions as a resource for generating arguments.

The views expressed in this section on legal knowledge sources are based on [53].

The subsuming category *legal writing* or legal document includes all writings that create a legally recognized fact. All writings required in a court procedure, in legislating, or in exercising a(nother) declarative or proclamative power (granting permits, taking administrative decisions, civil marriage ceremonies, etc.) are potentially legal resources, as well as writings required for compliance with duties to inform (tax application forms, etc.). Very often the writing explicitly includes an argumentation for the creation of the legally recognized fact: more generally it functions as evidence, backing, for its existence.

The legal knowledge source par excellence is the written *source of law*. The source of law is a writing that can be, is, was, or presumably will be used to back an argument concerning the existence of a rule in a certain legal system.

Alternatively, it is a writing used by a competent legislator to communicate a norm to a certain group of addressees. Both the legislator and the user of the source of law understand it as a medium used for communicating the existence of legal norms, as well as auxiliary declarations required for the proper understanding of the legal norms communicated between legislator and user, and *expect* that it can be used to back an argument concerning the existence of a norm or definition in a certain legal system.

Source of law is a familiar concept in law schools, and may be used to refer to both legislators (fonti delle leggi, sources des lois), legislation and case law (fonti del diritto, sources du droit), custom, etc. It should be noted that many romance languages make a distinction between the legislator as source of law, by way of speaking or writing, and the written law as source of right(s)[17].

In its broadest sense, the source of law is anything that can be conceived of as the originator of legal norms: for the purposes of this work it strictly refers to communication in writing. There are two main categories of source of law in writing: legislation and case law.

How case law functions as a source of law is addressed in section 6.7.2. Important to note here is that case law as a source of law posits general rules that can be applied to cases, just like legislation. It is not the embodied decision itself that functions as a source of law, but it explicit generalization to a more general rule, i.e. the argument ϕ because ψ is generalized to a rule ϕ if ψ.

The related, but disjoint, notion of a *legislative resource* includes all writings produced by the legislator explaining and justifying legislation. The legislator is a legal

[17] Also the Dutch *rechtsbron* is also more accurately translated as source of right.

person: it exists separately from any natural persons and organizations involved in the process of drafting and evaluating legislation. It is the formally correct completion of certain processes, usually dictated by law, that makes the legislator the formal author of a writing, and at the same time identifies the addressees to whom it applies. Obviously, the persons and organizations involved in the process of legislating may produce writings that are clearly precursors or legally required ingredients to the end product. These writings are also included in the notion of a legislative resource, but in this case it is not easy to give straightforward rules for deciding whether they are, or are not to be considered legislative resources. Different jurisdictions will have different theories on this subject.

As a last resort one can look at commentaries, written by independent parties.

Legal knowledge representation aims at an isomorphic representation of the source (cf. [17]), and isomorphism between the results of problem solving and existing written legal argument. The user who consults a system to determine the legality of some situation or action will accept the reference to the source as a backing for an argument if the relation between the argument and source is clarified by reading the source. A finer granularity level is usually better, but if the fragments of text get very small they loose their discourse context. The required granularity of isomorphism depends on the purpose of the system.

As pointed out in [42], deciding on the lexical scope of terms encountered in sources of law is of central importance: in that work an extremely skeptical approach of first representing each separate sentence, and integrating them by way of equivalence statements was advocated. In section 5.3 this approach is worked out in more detail. For maintenance not only isomorphism but also transparent version management (of source and of representation) is essential.

For the epistemic aspect to knowledge representation, legal knowledge engineering can refer to legal theory wherever the sources leave gaps. Legal reasoning is regularly portrayed as especially hard or ill-understood (see for instance [142, 201]), but this is rather caused by the wealth of *written* information – also detailing the many conceptual complications one encounters – available to fill the epistemological gap, and not its scarcity.

Hordes of brilliant minds have shed their light upon the problems of justifying legal decision making over the centuries, for good reasons because the law requires such justification. Few such analyses are available about the reasoning behind medical diagnosis, elevator design, scheduling mail delivery, configuring mainframe computers, etc: here *tacit knowledge* must be uncovered by the knowledge engineer through verbal reporting and observation techniques (see e.g. [257] for an overview of techniques)[18].

Moles in [201] knocks down a straw man, when he criticizes the over-reliance of knowledge engineers on codified law: in at least as many cases commercial LKBS could have benefited from *more* attention for codified law. This criticism also fails to acknowledge *why* LKBS are deployed: we are usually dealing with routine decision

[18]This obviously does not mean that there is *no* tacit knowledge in law: the epistemological gap is "closed" when knowledge engineers know enough to implement a KBS. In the case of law it can be almost closed by reading law and academic literature alone. Knowledge engineers obviously also have tacit knowledge, some of which is shared with the people whose tacit knowledge is being uncovered.

processes, and customers are typically not very interested in LKBS that make very original decisions involving especially serendipitous legal reasoning.

3.5.2. Generic Purpose Knowledge

Some domain theories have been worked out in detail in the knowledge engineering field, and can be considered as a given. These qualify as reusable knowledge components, although most of the involved schemas would need to be redesigned to conform to the design principles in this book (with respect to abstraction and separation of autoepistemic reasoning from ontology, or the principles of good ontology design).

Generally speaking, the best worked out domain is *naive physics* (cf. [144]) or qualitative physics. This area includes coherent common sense theories of the time and change (cf. [3, 4]), movement and places (cf. [99]), behaviour of liquids (cf. [143]), etc.

Time and change – in relation to actions and intentions – is quite central to legal reasoning, and merits a separate discussion. This is considerably less the case for, for instance, law and liquids.

Movement and place are relevant, in the field of law known as *spatial planning*: the combination of norms and places has its own logic, but is usually not considered a field of law in its own right[19]. The author of this work worked previously on this subject (cf. [40, 45, 46, 41]), in the context of the DURP project on standardization of spatial planning information in XML. They have no central position in legal theory, except in matters of (territorial) jurisdiction, spatial planning, and traffic.

Objects specific to law are however rarely found in physical reality. Most of them exist in the mental domain, where few similar well-developed ontologies exist.

Since the source of law has a commanding position in the legal setting, the nature of documents as expressions of legal knowledge is obviously a core domain, keeping in mind the remark of Makinson in [195] quoted earlier. The *Functional Requirements for Bibliographic Records* (FRBR; cf. [245]) are standard in this domain. In this book, in chapter 5, I follow *MetaLex* practices (cf. [53, 41, 40, 38, 37, 36, 285, 48, 51, 52, 49]), which are i.a. based on FRBR.

Another important area is of course the epistemic domain itself, in chapter 2 (see section 2.2.2) referred to as *folk psychology*. With the domain of intentional things (cf. [248, 148]), things that exist because we intend them to exist, it is a part of the mental domain. Of special relevance is the field of decision theory (cf. generally [137]). Although there is a wealth of scientific literature in these areas, there are no *specific* recommended well-articulated and widely available ontologies of it[20].

Norms in law, which clearly also belong to the nonphysical domain, combine several different aspects: they are intentional objects that justify attaching institutional qualifications to things, and they are standards to compare against, i.e. they are evaluative, or express what is better according to the legislator, etc.

[19]The field is dominated by social geographers.

[20]The categories of the mental and the social do lead to a lot of discussion in the development of *top ontologies*, and [202] for instance obviously includes many relevant categories in his legal epistemology.

Betterness appears to be a mental analogy to position: up is better, down is worse (cf. [183]). In law the scale is usually thought of as a binary one, which distinguishes things into legal and illegal. This is an artifact of hoe the legislator expresses norms: the legislator does not express himself in scales – for instance by grading behaviour like a teacher – but in deontic expressions. The scale is however usually not binary from an ex ante point of view: one is considering alternative hypothetical actions to formulate a plan. From an ex post assessment point of view it often appears to be binary, and it is often enough to apply it in this way, although we do occasionally need consider the question whether a better alternative was really at hand when a decision was made.

A lot of the academic research in this area is unfortunately under the heading of *preference*: this makes it conceptually a bit hard to consider norms as expressions of *betterness*. Amongst many others the following works have tried to explain norms in terms of preference relations: [138, 268, 262, 43, 44, 89].

3.5.3. Action, Change, and Intentions

Since norms in law can only be explained in terms of intention and decision making (from an *ex ante* point of view) or action and change (from an *ex post* point of view), theories of norms in law cannot be strictly separated from a theory of intention, action, and change. A lot of secondary legal terminology (right, liability, etc.) must be explained in terms of norms and action. In chapter 4 this combined theory will be developed.

Action brings together change and intention. If we consider events to be interesting changes, then actions are whatever it is that relates our intentions and events, honouring Wittgenstein's adagium (in [288]) that the act is in "what is left over if I subtract the fact that my arm goes up from the fact that I raise my arm."

Allen's general theory of action and time (cf. [3, 4]) and the event calculus (cf. [178]) are more or less standard in knowledge engineering, and have often been applied in legal knowledge engineering work. In addition, [151] specifies a standard way of representing time-indexed information in OWL.

The event calculus conceives of events as initiating and terminating fluents, and actions as events. One could say that initiation and termination focus attention to one half of a change. The initiation of one thing is the termination of something else. Event calculus includes frame assumptions that spell out what remains the same unless an event changing it is noted.

The theory focuses our attention on interesting changes against a backdrop of stasis, i.e. it is a typical foreground/background conceptualization, which depend on *granularity*. One could for instance describe the tidal cycle as rising tide (stasis), which turns (event) to falling tide (stasis), which again turns (event) to rising tide (stasis). Alternatively one could describe it as high tide (stasis), which falls (event) to low tide (stasis), and again rises (event) to high tide (stasis). Both are undoubtedly dreadful from a tidal analysis point of view, but a mention of the tide in a legal case will very likely pick one of these two perspectives. What the events are and what stasis is depends on what we are interested in (cf. generally [86]).

The law may obviously choose different levels of granularity, which may be incompatible (i.e. they don't reduce to each other), and one should respect the original formulation. Note that both events and stasis may take an unspecified amount of time, and both may be reduced into events and stasis at a lower level of granularity.

Our conception of what changes when has a large influence on how we reason about action, and about what ought to be done. Many logics of agency treat actions as instantaneous changes of state occurring at discrete time steps (i.e. *Hoare logic*, *dynamic modal logic*). It goes without argument that this is a superficial characterization of the nature of change (compared to [86]).

Its main feature is that this simple conception is computationally very powerful: the behaviour of computer programs is one of the domains for which this conceptualization works satisfactorily[21]. While even this theory is undoubtedly useful for legal knowledge engineering in some cases, in particular on the epistemic level, as a description of reasoning itself, one should be very wary of the conclusions one can reach using this theory about other domains, in particular the lawfulness of actions, because it embeds an important and very questionable assumption, as noted in section 4.7: the logic, instead of the law, may dictate whenever the *deadlines* occur.

The combination of Hoare-style logics with deontic logics of norms is an excellent example of the risks involved in mixing the autoepistemic assumptions of an abstract theory of one domain with those of another. This point will be discussed in more detail in section 4.7.

If we assume that the changes brought about by action are always intended, intention plays only a minor role. Although all actions are intentional, i.e. intend to achieve some change, they may however also fail to do so, or cause unforeseen side effects. Since the law distinguishes between intent, negligence, recklessness, etc. we do as a rule have to take into account the intentions of an agent, and we therefore have to commit to a "folk psychology" model of how the agent makes decisions.

Closely related to action and intention are social and institutional reality: these exist because they are intended to. These have been the subject of extensive philosophical debate (viz. for instance [248, 148, 164, 221]), but are very conspicuously glossed over – perhaps as too arbitrary to axiomatize them – by most ontologies. Institutional reality is one of the main subjects of chapter 4, and will be explained in terms of the constitutes and executes properties, that function as bridges between three different ontological strata.

3.6. Conclusions

Section 3.3 postulated the possibility of using knowledge components. If such a knowledge component represents the kind of knowledge that can be represented in OWL DL we take a *reasoning as process*, or *design* stance towards it. This is the case for ontologies and for abstract theories.

OWL DL knowledge components can be composed into structures of components through the import mechanism. An OWL DL knowledge component does not interro-

[21] That is, it works as long as everything of interest runs by the same system clock.

gate other OWL DL knowledge components, but directly imports them, if the resulting *belief base* is consistent. Composability is in principle only restricted by the constraint that the result must be consistent. From the point of view of impact analysis however, some import trees are better than others: For the purposes of this book, the representation of a source of law does not import generic purpose knowledge beyond the properties and concepts identified in this book for that purpose.

An OWL DL knowledge source is created by importing the OWL DL knowledge component into a conformant reasoner with a standardized query interface. A peculiarity of common query interfaces, like SPARQL, is that they permit an autoepistemic interpretation of the OWL DL knowledge source.

The interrogation of OWL knowledge bases by a Datalog-like rule language (cf. section 3.4.1) through a standard SPARQL query interface is a common setup for KBS. Because the rule language is not integrated into the semantics of the description logic, the resulting formal system is hard to formalize; Section 3.4.2 argued that this setup performs the essential function of autoepistemic reasoning. Three important such functions are enforcing constraints, inferring negation from failure, and applying default rules. Since there is a large variety of accounts for how and when such rules must be applied, we take a reasoning as decision making stance to it, and do not commit to an approach. The rule can be conceived of as a plan operator, to be used as part of a plan decided on in advance. In principle, it is possible to implement such rules in Datalog, if properly stratified through for instance the use of problem solving methods. It is also possible to use generic approaches to applying defeasible rules; Section 3.4.2 pointed out that Reiter's default logic is for instance implemented as an interface to OWL.

This chapter introduced two types of rules, the Default and the Constraint, whose semantics as a database update to the OWL DL knowledge component is relatively well-understood when considered in isolation, that will be used in this book. A set of these rules is a defeasible extension to a knowledge component. The knowledge component thus consists of:

1. a set of terminological axioms in OWL DL; and
2. a set of default and constraint rules.

A general principle is that it is only through the defeasible extension that one interrogates description logic knowledge components, that only the defeasible extension is used to drive interaction with human users, and that description logic components only import description logic components.

The focus of this work is the representation of sources of law as knowledge components. For the purposes of LKBS maintenance, one ideally maintains a clear mapping between knowledge components and the sources of law they are based on. The source of law is a writing that can be, is, was, or presumably will be used to back an argument concerning the existence of a rule in a certain legal system (cf. section 3.5.1).

In the next chapters some things will be taken for granted. There is no reason to dwell on the decomposability of things into their parts, or what effects this has in terms of inheritance of properties and relations from part to whole or vice versa. If article 2 of act a is changed, then a is also changed. If the date of publication of a is

t, then the date of publication of article 2 of act a is also t, etc. Time and space will also be taken for granted. Section 3.5.2 lists a number of these very general knowledge domains, that are so generic that it serves little purpose for this work to dwell on their complexities if it is of only ephemeral interest to interpretation of sources of law. Only the essentials, as proposed in the next two chapters, matter.

To bridge *ontological* strata the constitutes and executes property will be regularly used in this book, for reasons explained in chapter 4. The presence of certain properties tells us that a logical formula is not an ontological one. The major purpose of the use of *ontological stratification* in the following chapters is to methodologically align a content-based criterium – what does a the rule say? – with a technical concern – is the rule defeasible?