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°: 1.1 summary

Specification of the Legal Knowledge Interchange Format

Summary

Version:	1.0
Due date of Deliverable:	n.a.
Actual submission date:	February 1st, 2008
Start date of Project:	1 January 2006
Duration:	30 months
Project Coordinator:	Universiteit van Amsterdam (NL)
Lead contractor deliverable:	Universiteit van Amsterdam
Participating contractors:	Fraunhofer FOKUS
	Corvinus University Budapest

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)	

Specification of the Legal Knowledge Interchange Format

Deliverable 1.1: Summary

Alexander Boer, Marcello Di Bello, Kasper van den Berg
University of Amsterdam

Tom Gordon
Fraunhofer FOKUS

András Förhécz, Réka Vas
Corvinus University Budapest

Editors of the summary:

Szymon Klarman, Rinke Hoekstra
University of Amsterdam

Legal Knowledge Interchange Format (LKIF), being developed under the ESTRELLA project, is a Semantic Web based language for representing legal knowledge in order to support modeling of legal domains and to facilitate interchange between legal knowledge-based systems. In this summary we restate main points reported in the Deliverable 1.1: *Specification of the Legal Knowledge Interchange Format*. The objectives of the work package and requirements with respect to LKIF are recapitulated, followed by an overview of the language and technical specification of its constituent fragments. Additionally, the summary accounts for certain refinements in the formalization of LKIF rules, which have been accepted since the submission of the Deliverable.

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 editor:
Szymon Klarman
szymon@leibnizcenter.org

1 Introduction

The key goal of the ESTRELLA project is to develop a Legal Knowledge Interchange Format (LKIF), a language for supporting a machine accessible representation of knowledge in legal domains and providing a uniform open-standards based platform facilitating interchange between major proprietary formats already existing in the market of legal knowledge systems.

Deliverable 1.1: *Specification of the Legal Knowledge Interchange Format*, [Boer et al., 2007], submitted for reviewing to the ESTRELLA Consortium on 22/01 /2007, reported on the syntax and semantics of the first version of LKIF language and its component sublanguages. The technical part of the presentation was accompanied by a broad discussion of design choices and further possibilities arising from an overview of existing standards and technologies that were considered as relevant for the project. With this document we aim to provide a concise summary of the contents of D1.1, restricting ourselves only to issues having an essential bearing on the understanding of the characteristics of LKIF. It is therefore intended for an audience who is interested in getting a direct and substantial insight into the general architecture, core features and formal foundations of LKIF.

In section 2 we recall main goals of the work package and other more specific requirements that were directing the development of LKIF. In the following part we overview the structure of LKIF by discriminating between its three expressive dimensions and explaining main design choices made with respect to each of them. The dimensions correspond to three types of knowledge that obtain explicit support by particular fragments of LKIF language:

- *terminological knowledge* — supported in LKIF through the family of OWL languages (OWL Lite, OWL DL, OWL Full) and the LKIF-Core ontology of basic legal concepts, designed as another part of the work package,
- *legal rules* — represented by means of LKIF rules — a partly novel rule language developed specifically to account for argumentation-theoretic aspects of rules in legal domains,
- *normative statements* — handled on the level of terminological knowledge by a set of predefined deontic terms concentrated around the concept of subjunctive betterness.

Finally, in sections 4, 5 and 6 we present the syntax and semantics of the formal languages underlying respective dimensions, along with some illustrative examples of their practical applications.

It has to be noted that due to the ongoing progress on ESTRELLA project some claims made in D1.1 have clearly become outdated or would at least require a different qualification. A few tentative proposals for future developments have not eventually found their way to later versions of LKIF, whereas some others have been refined and received a much firmer standing in the current language. Repeating them here in the original form could hence contradict more recent documentation on LKIF and lead to unnecessary confusion as to its actual status. For that reason, whenever

it is justified, literal fidelity to the content of D1.1 gives way to conformity to the current state of knowledge. In any case, significant departures from the original text are explicitly signaled to the reader.

2 Objectives and Requirements

In this section we point out and briefly comment on main constraints imposed on LKIF upon the agreement by the participants of ESTRELLA. Listed objectives and requirements have been guiding the design process and remain the basis for justifying the resulting structure of the language.

2.1 Main Objectives

The general provision for work package 1 has been devised in Technical Annex to ESTRELLA. The following is the relevant excerpt from the document, [ESTRELLA, 2006, p. 16]:

> The main technical objectives of this work package are to develop a first version of a Legal Knowledge Interchange Format (LKIF), building upon emerging XML-based standards of the Semantic Web, including RDF and OWL, and Application Program Interfaces (APIs) for interacting with legal knowledge systems. The LKIF will apply the state of the art in the field of Artificial Intelligence and Law, taking into account business and application requirements. Existing Semantic Web initiatives are aimed at modelling concepts (OWL "ontologies") and rules (RuleML and SWRL). The LKIF will build on but go beyond this generic work to allow further kinds of legal knowledge to be modelled, including: meta-level rules for reasoning about rule priorities and exceptions, legal arguments, cases and case factors, values and principles, and legal procedures. In addition, an OWL ontology of basic legal concepts, such as obligations, permissions, rights and powers, will be developed, which can be reused when modelling a specific legal domain, such as tax law.

First of all, Technical Annex strongly emphasizes that the intended format should comply to the efforts for standardization of the Semantic Web technology. This requirement can be marked as one of the two of major importance for the project.

Semantic Web Compatibility: LKIF should be compatible with the emerging XML-based standards of the Semantic Web allowing for modeling concepts and rules. Consequently, it should take into account the underlying philosophy of reusability and interchangeability of information, modularity and layering.

Compatibility with the Semantic Web is for several reasons a desirable and practical requisite. Predominantly, it allows for saving considerable amounts of work on developing solutions that have been already provided, and in many respects proven satisfying, by the Semantic Web community. Moreover, it gives a reasonable guarantee that the architecture of LKIF will stay in lines with the main stream technologies and anticipated tools that will presumably follow the same standards.

Nevertheless, the technical objective might be relaxed to the extent necessary to secure achievement of the primary goal of the work package, which can be stated as follows:

Legal-Oriented Expressiveness: LKIF has to be a knowledge representation and interchange formalism, whose expressiveness is geared particularly to the demands of its intended users in the broadly understood market of legal knowledge systems.

What follows, LKIF has to be a domain-specific format, in the sense of being calculated for applications in the field of law, and a task-independent one, meaning that within that field it has to provide a universal support for tasks, independently from the choice of particular legal domain.

2.2 Requirements

In order to derive more specific requirements regarding LKIF some elaboration on the general objectives was necessary. This has been done by referring to three sources:

- a survey of research on computational models of legal reasoning and argumentation, from the field of Artificial Intelligence and Law;
- an analysis of the business requirements articulated by the participating vendors, and from the logical reconstructions of the logics used by these vendors, as reported in the companion document, Deliverable 1.2 [van den Berg et al., 2006];
- feedback and comments by the participating user organizations and members of the observatory board on earlier versions of the report.

The list of requirements included below is necessarily condensed for the purpose of this document, and does not exhaust all provisions developed in the research and discussed in D1.1[1]. Nonetheless, it should provide the reader with a detailed enough picture of the design principles to understand the actual architecture of LKIF, which will be introduced in the following sections. For the sake of clarity we roughly organize the presentation around technical and content-specific issues, though it has to be remembered that it is impossible to draw a firm line between them, as ones in a natural way constrain the others.

2.2.1 Technical Requirements

Further technical requirements account for basic logical, computational, and expressive properties of the language. Their motivation comes predominantly from the first principles and a general practice in Knowledge Representation and Semantic Web.

[1]The overview follows closely the elaboration of the requirements with respect to languages for modeling legislation presented in [Di Bello and van den Berg, 2007].

Requirement 1 LKIF has to be based on a logic equipped with a formally defined syntax and semantics.

Obviously, for providing an unambiguous representation of knowledge one needs a language operating exclusively with constructs of precisely defined meaning. It should be also clear according to what rules one should build proper expressions in that language so that they could convey the intended meanings. These two features — a formal specification of semantics and syntax — is clearly displayed by logical languages, what grants them a role of limiting constraints for knowledge representation formalisms.

Another apparent benefit from founding a representation and interchange format on a logical language is that a calculus associated with that language determines how to derive correct conclusions from well-formed knowledge bases, thus enabling particular reasoning tasks to be performed. Reversely, it specifies whether some set of expressions is sufficient to obtain a given information, what allows for optimizing interchange procedures by restricting the transfer only to minimum amount of information that is necessary in order to derive all the remaining, implicit knowledge.

Requirement 2 To the extent that automated reasoning services should be supported the expressiveness of LKIF (or its designated fragments) has to be confined to the boundaries of computational tractability.

The issue of computational tractability, though not of a primary importance for LKIF, is essential as far as application of automated reasoning techniques is considered. Roughly, the language cannot be too complex or expressive, otherwise automated reasoning becomes intractable. For formalisms based on First Order Logic a well-known limitation is decidability.

In case automated reasoning techniques are to be employed in the context of LKIF knowledge bases, it should be guaranteed that at least some of the fragments of LKIF maintain tractability, or that there exists a variant of representation preserving such fragments.

Requirement 3 The language should account for the layer of terminological knowledge and layer of rules, allowing for their separate treatment in the representation.

This requirement follows both from epistemological considerations on the nature of knowledge as well as from a good practice in Knowledge Representation and Semantic Web. Set of terminological definitions, denoted as ontology, usually expresses more entrenched parts of knowledge than ones embodied in rules, accounting for more operational level, to a bigger extent amenable to revision.

Semantic Web philosophy stresses the importance of layered and modularized approach to representation, which is supposed to ease interchangeability and reusability of information. Especially in legal knowledge representation, separating terminological knowledge from rules is a key factor facilitating reusability. While otologies represent common conceptual frame used for expressing world knowledge about

domains of applicability of law, and thus can be beneficially exchanged between different jurisdictions, normative knowledge, expressed in rules, may vary between legislative acts and systems, hence its reusability is usually of a secondary interest.

Complying to the Semantic Web standards, LKIF would adopt the postulate of layering and separation of different types of knowledge naturally.

Requirement 4 LKIF should enable, to the possible extent, interchange between formats already used by vendors in the legal market.

As an interchange format, LKIF should enable translations between proprietary formats with none or minimal loss of information. The requirement entails that the language should be sufficiently expressive to embrace expressivity of other formats, and so it is clearly traded off against others (such as 2), insisting on restricting, rather than increasing expressiveness of LKIF. Moreover it might turn out that it is not possible at all to find a common denominator for different formats if the discrepancies between underlying logics are too significant. In fact, D1.2 [van den Berg et al., 2006] has made it clear that this is the case. As a fallback solution, LKIF should be supplemented with translators that are able to export exact amounts of information expressed in a vendor's format that are meaningful in LKIF, and vice versa.

2.2.2 Expressiveness Requirements

The requirements from this part reflect most important legal domain-specific demands for LKIF.

Requirement 5 The language should provide some form of support for handling exceptions to norms.

The notion of exception is inherent to legal domain and normative reasoning. Roughly, there are two different approaches to resolving normative conflicts, which are due to other reasons than mere inconsistency of law. One way requires extending the language with an additional operator, e.g. *unless*, for explicit modeling of exceptions within the scope of rules. The other option is to resort to meta-rules, such as *lex superior*, *lex posterior* and *lex specialis*, for adjudicating between priorities of conflicting norms.

Both solutions have some clear advantages and drawbacks. The first one leads to increasing the expressiveness of the language, what may affect its computational tractability. Moreover exceptions are quite often implicit to legislation, and thus can be hard to detect by a modeler. The second approach requires an external system to recognize exceptions and apply appropriate meta-rules.

Requirement 6 Rules have to be given a two-fold representation in the language: as formulae constructed of terms, atoms and operators, and as objects with properties.

In many legal contexts normative rules appear as objects with properties and relations to other objects. Example properties include various dates, such as the

date of enactment, period of validity and date of repeal, authority which issued the rule, the backing of the rule (i.e. a reference to the legal source) or issues concerning applicability of the rule. Hence, representing norms exclusively as logical formulae does not allow for an adequate reconstruction of the model of a legislation. In addition to that, rules have to be also *reified* in LKIF.

Requirement 7 An LKIF knowledge base needs to provide a representation of all the information needed for applying the argumentation schemes to be supported.

The requirement follows from a broader perspective on the nature of law. Evidently, there is much more to legal reasoning than just representation of norms in terms of logical formulae and application of deductive rules of inference. Such a naive picture, popularly denoted as *mechanical jurisprudence* [Rissland et al., 2003], has been collectively rejected by theorists of law. Instead, the essential role in legal reasoning is assigned to argumentation, i.e. a process of determining whether facts of a case can be "subsumed" under some legal concept.

Each legal argumentation task operates on argumentation schemes, which generalize the concept of an inference rule to cover plausible as well as deductive and inductive forms of argument. Although LKIF is not primarily intended to support such forms of reasoning, it is at least expected to be an expressive enough language to represent all the information that might be required for such inferences by an external engine.

Requirement 8 LKIF should contain some set of deontic concepts for expressing normative statements, based on subjunctive betterness of things described by the concept.

Deontic operators certainly belong to the most law-specific part of the vocabulary used for expressing legal knowledge. Such terms as prohibition, permission, obligation are indispensable for attributing a normative power to otherwise descriptive statements. The central concept to the field of deontic reasoning is *subjunctive betterness*, which carries a fundamental deontic intuition involved in statements like: α is better than $\neg\alpha$.

LKIF should not incorporate any particular variant of deontic logic, while this would unnecessarily impose restrictions on possible applications. In particular some seemingly natural constraints (e.g. deontic conflicts should entail contradiction) should be given in LKIF a special, more lenient treatment than it is sanctioned in typical deontic logics. Still, it should account for basic deontic constructs and their core properties entailed by the very meaning of the concept of subjunctive betterness, so that modeling of various deontic systems is directly supported.

3 Overview of LKIF

LKIF has been designed as a knowledge interchange format targeted particularly for applications in legal domains. Its architecture is a resulting compromise between a range of objectives and requirements, which in many cases could not have been satisfied simultaneously. The dominant trade-off underlying the design process concerned conflicting demands for an increased expressiveness of the language on the one hand, and its tractability and compatibility with established standards on the other. In this section we present a brief overview of the structure of LKIF and specify its fragments, which will be addressed in the following sections.

In general, LKIF provides a direct support for representing three types of knowledge, which have been identified as most indispensable to law and legal reasoning: *terminological knowledge*, *legal rules* and *normative statements*.

Following the adopted requirements, LKIF complies to Semantic Web and Knowledge Representation philosophy of a layered representational structure. Predominantly, we can distinguish between terminological and rule layer of the language, which constitute its two expressive dimensions. Furthermore, LKIF vocabulary reserves a special set of deontic concepts aimed for modeling normative statements.

Terminological Knowledge: The layer of terminological knowledge is supported in LKIF through the Web Ontology Language (OWL), one of the recommended XML-based standards for Semantic Web. OWL allows for explicit and formal representation of meaning of terms and relations between them, thus enabling construction of machine-accessible models of ontologies underlying particular domains of knowledge. Ontologies specified in OWL can be stored as OWL files.

Several variants of OWL and associated formalisms have been considered for the purpose of representing terminological knowledge for legal applications:

OWL Lite The least expressive sublanguage of OWL, meant mostly for representing classification hierarchies and simple constraints. The limited expressiveness of OWL Lite effects in a low computational complexity.

OWL DL Variant of OWL closely related to Description Logic, providing maximum expressiveness while retaining computational completeness and decidability (all conclusions are guaranteed to be computable in finite time).

OWL Full The most expressive sublanguage, allowing for syntactic freedom going beyond DL, however with no computational guarantees — OWL Full falls into an undecidable subset of First Order Logic.

OWL DLP Intersection of OWL DL and a subset of logic programs. A strict subset of OWL with attractive computational properties and a strong support from existing reasoning tools, though with a somewhat limited expressiveness.

OWL DL + SWRL A combination of OWL DL and Semantic Web Rule Language. A variant granting very high expressiveness at the expense of computational complexity and heterogenous architecture.

The variant that seems to present best overall value, taking into account its expressiveness, computational properties and potential applications, and which thus has been primarily employed in the terminological layer of LKIF is OWL DL.

OWL DLP can also be seen as a promising candidate, especially that obtaining this fragment of OWL requires merely some rigor in the way of using certain OWL constructs. However, the benefit from making a relatively small shift towards computationally more efficient, but less expressive OWL DLP, can only be assessed in practical setting, and most likely should be task-dependent.

On the contrary, considering prospective tasks to be supported by LKIF, OWL Lite and OWL Full both do not manage to balance properly the expressiveness-tractability trade-off, whereas combination of OWL and SWRL looses its practicality in the light of adopting another, even more expressive rule formalism. In any case LKIF rules can compensate for lack of expressiveness on the terminological layer.

The terminological support provided by LKIF is further extended with the LKIF-Core ontology of basic legal concepts, designed as another part of the work package. The LKIF ontology consists of the following modules, available as different OWL ontology files:

Expression	covers a number of representational primitives necessary for describing relational mental states that connect a person to a proposition: e.g. beliefs, statements etc.
Processes	describes concepts related to (involuntary) change;
Action	covers concepts related to actions and their relation to physical change and intentions, e.g. action, agent etc;
Role	describes constructs that underlie the roles being played by agents;
Place	defines representational primitives for describing places, locations and the relations between them;
Time	covers representational primitives for describing time intervals
Mereology	describes classes and properties that allow to express mereological relations, e.g. parthood, components etc.
Argumentation	describes concepts central to LKIF argumentation, and the components of LKIF Rules: e.g. assumption, exception, rule;
Norm	describes the concepts most central to LKIF: e.g. norm, obligation, prohibition etc.

These are considered the necessary components for explaining the types of frameworks, whether epistemological, situational, or mereological that are common in law. Typical legal concepts (like burden of proof, potestative right, administrative ap-

peal) can be described in terms of them, but are not themselves the focus of the work on LKIF.

Legal Rules: Whereas terminological layer has easily fit into the recommended Semantic Web standards, representation of legal rules has required more sophisticated formalism. Initially, one serious candidate language has been considered:

SWRL — *Semantic Web Rule Language* — put forward as a proposal for Semantic Web rules-language, combining sublanguages of the OWL Web Ontology Language (OWL DL and Lite) with those of the Rule Markup Language (Unary/Binary Datalog)

SWRL has turned out insufficiently expressive to capture some of the essential peculiarities of legal reasoning. For that reason, a partly novel rule formalism, called LKIF rules, has been developed and incorporated in LKIF.

LKIF rules extend SWRL with support for negation and defeasible reasoning. The rule layer provides a language expressive enough to model legal rules in a way which comes much closer to the ideal of *isomorphic modeling* [Bench-Capon and Coenen, 1992], i.e. in a way which reflects the structure of the rules in a legislation. The rules layer supports rules with exceptions, assumptions, and exclusionary conditions, and enables meta-level information about rules to be represented, such as date of enactment, which is used in other rules, such as *lex posterior*, to reason about rule priorities.

Argumentation module in LKIF-Core ontology provides a conceptual framework, underpinned by an argumentation-theoretic semantics, for representing rule constructs as OWL objects. Rules can be hence embedded in OWL files alongside terminologies.

It has to be noted that after completion of D1.1, W3C announced a proposal for *Rule Interchange Format*, a new Semantic Web standard for combining rules with ontologies. This formalism, though not originally considered, should be taken into account and assessed from the perspective of a possible reorientation of the LKIF rule layer in the future. For this reason we shall give some attention to RIF in section 5.

Normative Statements: Normative statements are given a direct support via *Norm* module included in LKIF-Core ontology. No particular deontic logic is imposed on LKIF representation. Such an approach is justified by both theoretical and practical reasons. On the one hand there is a little convergence in the field of deontic logic to the standard system of reasoning, on the other — industrial applications do not typically require a reference to any concrete deontic logic. Instead, the module contains minimally restricted definitions of deontic concepts and properties, based on the notion of subjunctive betterness, which are intended as basic constructs for modeling various deontic settings. Nevertheless, there exists a preferable mapping of normative statements into OWL representation, which preserves some essential properties of the modal frames semantics, associated with deontic logics.

To summarize, Legal Knowledge Interchange Format is an OWL ontology of legal concepts allowing legal knowledge bases, encompassing specific terminologies, LKIF rules, and normative statements, to be represented in OWL and stored as OWL files. Let us finally introduce the notion of an LKIF reasoner, an LKIF file and of using LKIF:

Definition 1 (Using LKIF) *To use LKIF is to refer to an entity in the LKIF namespace.*

Definition 2 (LKIF File) *An LKIF file is a file which uses one of the concrete syntaxes described in this section, is limited to one of the expressive fragments identified in this section, and refers to an entity in the LKIF namespace.*

Definition 3 (LKIF Reasoner) *An LKIF reasoner is a reasoner that interprets a set of RDF triples that refer to an LKIF entity, and that is limited to one or more of the identified expressive fragments, in accordance with LKIF semantics.*

In the following sections we will focus on the formal specification of the three basic fragments of LKIF singled out in this overview: OWL DL, LKIF rules and deontic concepts in *Norm* module.

4 Terminological Layer

OWL DL is one of the Semantic Web languages recommended for ontological modeling, i.e. for representing semantic structure of the terminology underlying a domain of application and for expressing knowledge about individuals in the domain by means of that terminology. Among the features that especially bore upon employing OWL DL in LKIF most significant are: firm logical underpinning, decidability, compatibility with XML-based standards, a good support by existing tools and growing popularity in web applications.

Recently, another version of ontology language — OWL 1.1 — has been proposed by W3C. OWL 1.1 adds several new features to OWL DL, and thus might take over the role of OWL DL in the subsequent versions of LKIF. We address this issue at the end of the section.

4.1 Formal Foundations of OWL DL

OWL DL is a formalism emerged at the intersection of several lines of research in Semantic Web and Knowledge Representation. In this section we will mention two languages that in different ways influenced design of OWL DL and point out some of their characteristics that particularly shaped the structure and properties of OWL DL[2].

4.1.1 RDF/RDFS

Resource Description Framework (RDF) is a metadata model developed by W3C. The central idea behind is to establish a uniform representation for various statements that can be made about entities (resources) in an application domain. The representation has a form of an *RDF triple*:

$$\langle subject, predicate, object \rangle$$

where *subject* carries a reference to a resource and *predicate* denotes a relation that holds between the subject and some other resource referred by *object*.

The *RDF vocabulary* is a set of RDF names. An *RDF name* is a *Uniform Resource Identifier* reference or a *literal*. A URI can be a *Uniform Resource Locator* or *Uniform Resource Name*, which are both registered strings of symbols that unambiguously identify an entity. Literals are lexical expressions representing some data values. URI names can occur as subjects, predicates or objects of RDF triples, whereas literals only as objects. A set of RDF triples is represented by an *RDF graph*, whose nodes are either labeled with RDF names or remain unnamed as so-called *blank nodes*. An RDF graph can be completely expressed in *XML syntax*, thus it preserves compatibility with the older family of markup languages.

An RDF interpretation is a tuple $\langle \mathbb{R}, \mathbb{P}, I_L, I_{URI}, I_P, g \rangle$, where $\mathbb{R}$ is a domain of resources, $\mathbb{P}$ is a set of properties and the remaining are interpretation functions defined as follows:

[2]For a complete presentation of RDF/S see `http://www.w3.org/RDF/` and `http://www.w3.org/TR/rdf-schema/`.

- $I_L : L \mapsto \mathbb{R}$;
- $I_{URI} : URI \mapsto \mathbb{R} \cup \mathbb{P}$;
- $I_P : \mathbb{P} \mapsto \wp(\mathbb{R} \times \mathbb{R})$;
- $g : B \mapsto \mathbb{R}$.

Summarizing, I_L provides meanings for all literals, I_{URI} for URI's, some of which are mapped to the domain of resources, and some to the domain of properties, I_P assigns a set of resource pairs to every property, whereas g interprets all blank nodes in the RDF graph. A very characteristic feature of RDF semantics is the distinction between intensional and extensional level of interpretation. Notice that some URI's are first assigned an object from $\mathbb{P}$ (intension) and then a subset of $\mathbb{R} \times \mathbb{R}$ (extension).

RDF is equipped with a set of predefined URI references, which due to basic axiomatization, provide an initial vocabulary for extending the language and constructing complex descriptions about resources. Elements of this vocabulary are e.g. `rdf:type`, `rdf:property`, `rdf:Statement`, `rdf:subject`, `rdf:predicate`, `rdf:object`.

RDF Schema (RDFS) is an extension to RDF, which adds further constructs to the RDF specific vocabulary, e.g.: `rdfs:domain, rdfs:range, rdf:Resource, rdfs:Class`, `rdfs:subClassOf`. Moreover it augments an interpretation of RDF graph with a domain of classes $\mathbb{C}$ and a respective interpretation function $I_C : \mathbb{C} \mapsto \wp(\mathbb{R})$, which maps classes to sets of resources. Consequently, the range of I_{URI} is extended to $\mathbb{R} \cup \mathbb{P} \cup \mathbb{C}$, hence duality between intensional and extensional level holds for classes in the same manner as for properties.

4.1.2 Description Logic

Description Logic (DL) is a family of logical languages intended as knowledge representation formalisms for expressing *terminological* and *assertional knowledge* about a domain of application, [Baader et. al, 2003]. The languages are classified according to their expressive power, defined in terms of constructors available for building complex concepts and roles. All well formed DL formulae be given a straightforward translation to First Order Logic (FOL), therefore every DL language corresponds to some subset of FOL and obtains a formal, model-theoretic semantics. A significant feature of this correspondence is that all DL languages remain in the decidable subset of FOL, and so they are computationally tractable.

Knowledge in DL is expressed in terms of concepts (FOL one place predicates) and roles (FOL two place predicates), which can be used for expressing statements about individuals in the domain. A language is defined by a set of atomic names for concepts, roles and individuals and a set of concept and role constructors. The semantics is given by an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is a non-empty domain of individuals and $\cdot^{\mathcal{I}}$ is an interpretation function defining the meaning of the vocabulary as follows:

- $\cdot^{\mathcal{I}} : N_I \mapsto \Delta^{\mathcal{I}}$ for individual names,
- $\cdot^{\mathcal{I}} : N_C \mapsto \wp(\Delta^{\mathcal{I}})$ for concept names,

- $\cdot^{\mathcal{I}} : N_R \mapsto \wp(\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}})$ for role names.

The reminder of semantics is defined inductively on the set of construction rules.

For the purpose of this presentation let us only mention the DL language of expressiveness denoted as $\mathcal{SHOIN}$. Table 1 presents the set of constructors that can be used in DL $\mathcal{SHOIN}$ for arbitrary concepts C, C_1, C_2, a role P, an individual i and a non-negative integer n, in order to obtain other concepts and roles.

Constructor	DL Syntax	FOL Syntax
concept negation	$\neg C$	$\neg C(x)$
concept intersection	$C_1 \sqcap C_2$	$C_1(x) \wedge C_2(x)$
concept union	$C_1 \sqcup C_2$	$C_1(x) \vee C_2(x)$
nominal	$\{i\}$	no counterpart
existential restriction	$\exists P.C$	$\exists y(P(x,y) \wedge C(y))$
universal restriction	$\forall P.C$	$\forall y(P(x,y) \rightarrow C(y))$
minimum cardinality	$\geq n\ P$	$\exists y_1, \ldots y_n P(x,y_1) \wedge \ldots \wedge P(x,y_n)$ $\rightarrow \bigwedge_{i<j}(y_i \neq y_j)$
maximum cardinality	$\leq n\ P$	$\forall y_1, \ldots y_{n+1} P(x,y_1) \wedge \ldots \wedge P(x,y_{n+1})$ $\rightarrow \bigvee_{i<j}(y_i = y_j)$
inverse role	P^-	no counterpart
transitive closure	P^+	no counterpart

Table 1: DL Concept and Role Constructors.

Furthermore, there are two special constructors used for formulating DL axioms, i.e. concept and role definitions and hierarchies (see Table 2).

Constructor	DL Syntax	FOL Syntax
concept equivalence	$C_1 \equiv C_2$	$C_1(x) \leftrightarrow C_2(x)$
concept hierarchy	$C_1 \sqsubseteq C_2$	$C_1(x) \rightarrow C_2(x)$
role equivalence	$P_1 \equiv P_2$	$P_1(x,y) \leftrightarrow P_2(x,y)$
role hierarchy	$P_1 \sqsubseteq P_2$	$P_1(x,y) \rightarrow P_2(x,y)$

Table 2: DL Axioms Constructors.

A set of axioms constitutes a TBox (terminological knowledge) of a DL knowledge base, whereas a set of assertions of the form $C(i)$ or $P(i_1, i_2)$ is its ABox (assertional knowledge). A knowledge base is *satisfiable* if it has a model, i.e. an interpretation in which all axioms and assertions are satisfied.

4.2 OWL DL

The expressiveness of OWL DL is denoted as $\mathcal{SHOIN}(\mathbf{D})$, which signals the extension of the standard DL $\mathcal{SHOIN}$ with datatypes. Even so, the logical foundations of OWL DL are fully determined by description logic. From RDF the language inherits considerable part of specific vocabulary (among others data types) and, above

all, syntactic compatibility with RDF triples and XML[3].

4.2.1 Syntax

The *OWL Vocabulary* is defined as $V ::= V_L \cup V_{URI}$, where V_L is a set of literals and V_{URI} a set of URI references. The set V_{URI} is composed as follows:

- V_I, set of individual names, e.g. `Peter`,
- V_O, set of ontology names, generally consisting of URL addresses indicating where ontologies are stored,
- V_{IC}, set of object class names, e.g. `owl:Thing`, `owl:Nothing`,
- V_{DC}, set of datatype class names, e.g. `rdfs:Literal`, `xsd:gDay`, `xsd:integer`,
- V_{IP}, set of object property names, e.g. `has_father`,
- V_{DP}, set of datatype property names, e.g. `height_in_meters`,
- V_{AP}, set of annotation property names, e.g. `owl:label`, `owl:seeAlso`,
- V_{OP}, set of ontology property names, e.g. `owl:import`.

Notice that OWL vocabulary covers all the types introduced by DL and RDF, and adds new ones, all interpreted as URI's. Notions of concept and role present in DL are replaced by their RDF counterparts: class and property. Additionally, the *OWL specific vocabulary* consists of symbols (e.g. `owl:intersectionOf`, `owl:inverseOf`, `owl:EquivalentClasses`) strictly corresponding to DL concept, role and axiom constructors. Whenever possible, in this overview we will use less verbose DL notation.

Definition 4 *The set of* OWL classes *and* OWL properties *are recursively defined by the following rules:*

$$\begin{aligned} \textit{OWL-Class} ::= & C \mid \top \mid \bot \mid \neg C \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \\ & \forall P.C \mid \exists P.C \mid \forall T.D \mid \exists T.D \mid \\ & \leq n P^{int} \mid \leq n T \mid \{i_1, \ldots, i_n\} \mid \{l_1, \ldots, l_n\} \\ \textit{OWL-Property} ::= & P \mid P^{-} \end{aligned}$$

where C stands for a class (atomic or complex); $P \in V_{IP}$ an atomic property, $D \in V_{DC}$ a datatype class; and $T \in V_{DP}$ a datatype property. The symbols $\top$ and $\bot$ are short for `owl:Thing` and `owl:Nothing`. P^{int} means that the property P is not a transitive property or is not a sub-property of a transitive property.[4]

[3] A comprehensive account of OWL syntax and semantics can be found on `http://www.w3.org/TR/owl-semantics/`.

[4] This is one of the expressive limitations of OWL DL.

The construction rules for *OWL axioms* follow DL syntax. Hence, OWL mirrors the distinction between *TBox* — set of equivalence and subsumption conditions on classes and properties — and *ABox* — set of type and property assertions. OWL ABox might moreover contain equivalence and non-equivalence conditions on individuals, which are introduced as a result of dropping the *Unique Name Assumption*, characteristic for DL. Several constructors are added (e.g. `owl:FunctionalProperty`, `owl:SymmetricProperty`, `owl:DisjointClasses`) as syntactic sugar meant to simplify notation of some useful axioms.

Definition 5 *The set of* OWL axioms *is defined by the following rules:*

$$\begin{aligned}
\mathcal{T}\text{-}Axiom ::= & C_1 \sqsubseteq C_2 \mid C_1 \equiv C_2 \mid P_1 \sqsubseteq P_2 \mid P_1 \equiv P_2 \mid \\
& D_1 \sqsubseteq D_2 \mid D_1 \equiv D_2 \mid T_1 \sqsubseteq T_2 \mid T_1 \equiv T_2 \\
& \textsc{Trans}(P) \mid \textsc{Func}(P) \mid \textsc{Symm}(P) \\
& \textsc{Disjoint}(C_1, C_2) \mid \textsc{Disjoint}(D_1, D_2) \\
\mathcal{A}\text{-}Axiom ::= & C(i) \mid P(i_1, i_2) \mid i_1 = i_2 \mid i_1 \neq i_2 \mid \\
& D(l) \mid T(l_1, l_2) \mid l_1 = l_2 \mid l_1 \neq l_2
\end{aligned}$$

Notice that datatype classes and properties cannot be part of many OWL expressions, otherwise available to object classes and properties. Strict distinction between the two categories is a warrant of decidability of OWL-DL.

4.2.2 Semantics

An *OWL model* $\mathfrak{M}_{OWL}$ is a tuple $\langle \mathbb{R}, \mathbb{R}_D, \mathbb{R}_O, I_C, I_P, I_I, I_L \rangle$. The set $\mathbb{R}$ is a domain of resources, with $\mathbb{R}_O \subseteq \mathbb{R}$ a set of objects or individuals, and $\mathbb{R}_D \subseteq \mathbb{R}$ a set of datatypes or literal values (i.e. either the literal itself, if it is a plain literal, or its proper value, if it is a typed literal). Notice that $\mathbb{R}_D \cap \mathbb{R}_O = \emptyset$. Each I_i is an interpretation function for classes, properties, individual names or literals. More precisely:

- $I_C(\texttt{owl:Thing}) = \mathbb{R}_O \subseteq \mathbb{R}$
- $I_C(\texttt{owl:Nothing}) = \{\} \subseteq \mathbb{R}$
- $I_C(\texttt{owl:Literal}) = \mathbb{R}_D \subseteq \mathbb{R}$
- $I_C : V_C \mapsto \wp(\mathbb{R}_O)$
- $I_C : V_D \mapsto \wp(\mathbb{R}_D)$
- $I_P : V_{DP} \mapsto \wp(\mathbb{R}_O \times \mathbb{R}_D)$
- $I_P : V_{IP} \mapsto \wp(\mathbb{R}_O \times \mathbb{R}_O)$
- $I_P : V_{AP} \cup \{\texttt{rdf:type}\} \mapsto \wp(\mathbb{R} \times \mathbb{R})$
- $I_P : V_{OP} \cup \{\texttt{rdf:type}\} \mapsto \wp(\mathbb{R} \times \mathbb{R})$

- $I_I : V_I \mapsto \mathbb{R}_O$
- $I_L : V_L \mapsto \mathbb{R}_D$

The above definition includes the semantics for all atomic classes and properties. The semantics for complex constructions is defined recursively by the rules presented in table 3.

OWL-Syntax	Semantics
$\neg C$	$\mathbb{R}_O \setminus I_C(C)$
$C_1 \sqcap C_2$	$I_C(C_1) \cap I_C(C_2)$
$C_1 \sqcup C_2$	$I_C(C_1) \cup I_C(C_2)$
$\forall P.C$	$\{o \in \mathbb{R}_O \mid (o, o') \in I_P(P) \Rightarrow o' \in I_C(C), \text{ for all } o'\}$
$\exists P.C$	$\{o \in \mathbb{R}_O \mid (o, o') \in I_P(P) \ \wedge \ o' \in I_C(C), \text{for some } o'\}$
$n \leq P$	$\{o \in \mathbb{R}_O \mid \#\{o' \mid (o, o') \in I_P(P)\} \leq n\}$
$\forall T.D$	$\{o \in \mathbb{R}_O \mid (o, d) \in I_P(T) \Rightarrow d \in I_C(D), \text{ for all } d\}$
$\exists T.D$	$\{o \in \mathbb{R}_O \mid (o, d) \in I_P(T) \ \wedge \ d \in I_C(D), \text{ for some } d\}$
$n \leq T$	$\{o \in \mathbb{R}_O \mid \#\{d \mid (o, d) \in I_P(T)\} \leq n\}$
$\{i_1, \ldots i_k\}$	$\{I_I(i_1), \ldots, I_I(i_k)\}$
$\{l_1, \ldots l_k\}$	$\{I_L(l_1), \ldots, I_L(l_k)\}$
P^-	$\{(o, o') \mid (o', o) \in I_P(P)\}$

Table 3: Semantics for OWL DL concept constructors.

In the final step we specify truth conditions for OWL axioms and ontologies.

Definition 6 *The* OWL truth *relation is defined for OWL axioms as follows:*

$$
\begin{array}{lll}
\mathfrak{M}_{OWL} \models C(i) & \text{iff} & I_I(i) \in I_C(C) \\
\mathfrak{M}_{OWL} \models C_1 \sqsubseteq C_2 & \text{iff} & I_C(C_1) \subseteq I_C(C_2) \\
\mathfrak{M}_{OWL} \models C_1 \equiv C_2 & \text{iff} & I_C(C_1) = I_C(C_2) \\
\mathfrak{M}_{OWL} \models P(i_1, i_2) & \text{iff} & (I_I(i_1), I_I(i_2)) \in I_P(P) \\
\mathfrak{M}_{OWL} \models P_1 \sqsubseteq P_2 & \text{iff} & I_P(P_1) \subseteq I_P(P_2) \\
\mathfrak{M}_{OWL} \models P_1 \equiv P_2 & \text{iff} & I_P(P_1) = I_P(P_2) \\
\mathfrak{M}_{OWL} \models i_1 = i_2 & \text{iff} & (I_I(i_1), I_I(i_2)) \in I_P(=) \\
\mathfrak{M}_{OWL} \models i_1 \neq i_2 & \text{iff} & (I_I(i_1), I_I(i_2)) \notin I_P(=)
\end{array}
$$

$$
\begin{array}{lll}
\mathfrak{M}_{OWL} \models D(l) & \text{iff} & I_L(l) \in I_C(D) \\
\mathfrak{M}_{OWL} \models D_1 \sqsubseteq D_2 & \text{iff} & I_C(D_1) \subseteq I_C(D_2) \\
\mathfrak{M}_{OWL} \models D_1 \equiv D_2 & \text{iff} & I_C(D_1) = I_C(D_2) \\
\mathfrak{M}_{OWL} \models T(l_1, l_2) & \text{iff} & (I_I(l_1), I_I(l_2)) \in I_P(T) \\
\mathfrak{M}_{OWL} \models T_1 \sqsubseteq T_2 & \text{iff} & I_P(T_1) \subseteq I_P(T_2) \\
\mathfrak{M}_{OWL} \models T_1 \equiv T_2 & \text{iff} & I_P(T_1) = I_P(T_2) \\
\mathfrak{M}_{OWL} \models l_1 = l_2 & \text{iff} & (I_L(l_1), I_L(l_2)) \in I_T(=) \\
\mathfrak{M}_{OWL} \models l_1 \neq l_2 & \text{iff} & (I_L(l_1), I_L(l_2)) \notin I_T(=)
\end{array}
$$

$$
\begin{array}{lll}
\mathfrak{M}_{OWL} \models \textsc{Disjoint}(C_1, C_2) & \text{iff} & I_C(C_1) \cap I_C(C_2) = \emptyset \\
\mathfrak{M}_{OWL} \models \textsc{Trans}(P) & \text{iff} & (o_1, o_2) \in I_P(P) \wedge (o_2, o_3) \in I_P(P) \Rightarrow \\
 & & \Rightarrow (o_1, o_3) \in I_P(P) \text{ for all } o_i \in \mathbb{R}_O \\
\mathfrak{M}_{OWL} \models \textsc{Func}(P) & \text{iff} & (o_1, o_2) \in I_P(P) \wedge (o_1, o_3) \in I_P(P) \Rightarrow \\
 & & \Rightarrow o_2 = o_3 \text{ for all } o_i \in \mathbb{R}_O \\
\mathfrak{M}_{OWL} \models \textsc{Symm}(P) & \text{iff} & (o_1, o_2) \in I_P(P) \Leftrightarrow (o_2, o_1) \in I_P(P) \\
 & & \text{for all } o_i \in \mathbb{R}_O \\
\mathfrak{M}_{OWL} \models \textsc{Disjoint}(D_1, D_2) & \text{iff} & I_C(D_1) \cap I_C(D_2) = \emptyset
\end{array}
$$

Definition 7 *Given a vocabulary V, and an ontology $\mathcal{O} ::= \mathcal{T}$-axiom $\cup$ $\mathcal{A}$-axioms, ontology satisfaction, consistency and entailment are defined as follows:*

$$
\begin{array}{lll}
\mathfrak{M}_{OWL} \models \mathcal{O} & \text{iff} & \mathfrak{M}_{OWL} \models \varphi \text{ for all } \varphi \in \mathcal{O} \\
 & & \text{and any linguistic item in } \mathcal{O} \text{ is backed in } V \\
\mathcal{O} \nvdash \bot & \text{iff} & \mathfrak{M}_{OWL} \models \mathcal{O}, \text{for some } \mathfrak{M}_{OWL} \\
\mathcal{O} \models \mathcal{O}' & \text{iff} & \mathfrak{M}_{OWL} \models \mathcal{O} \Rightarrow \mathfrak{M}_{OWL} \models \mathcal{O}' \text{ for all } \mathfrak{M}_{OWL}
\end{array}
$$

The last definition accomplishes presentation of OWL semantics.

4.2.3 Examples

Several examples of how terminological knowledge can be expressed in OWL are presented below. All axioms are borrowed from the model of EU Council Directive of 23 July 1990 (31990L0434) being developed as a test case for LKIF.

$$
\begin{array}{lll}
\textsc{Transfer} & \sqsubseteq & \textsc{Action} \\
\textsc{Transfer} & \equiv & \exists \mathsf{actor}.(\textsc{Agent} \sqcap \\
 & & (\exists \mathsf{transfers}.(\textsc{Physical\_Object} \sqcup \textsc{Ownership})))
\end{array}
$$

$$
\begin{array}{lll}
\textsc{Asset} & \sqsubseteq & \textsc{Economic\_Role} \\
\textsc{Asset} & \equiv & \exists \mathsf{played\_by}.((\exists \mathsf{value}.\textsc{Money}) \sqcap (\exists \mathsf{owned\_by}.\textsc{Legal\_Person}))
\end{array}
$$

$$
\begin{array}{lll}
\textsc{Enterprise} & \sqsubseteq & \textsc{Legal\_Person} \\
\textsc{Enterprise} & \sqsubseteq & \textsc{Business} \sqcup \textsc{Company}
\end{array}
$$

4.3 OWL 1.1

The W3C has recently started a new working group to develop a successor of the OWL Web Ontology Language: OWL 1.1[5]. This language extends the original OWL with a number of features for which effective reasoning algorithms are available, and which meet real user needs. Until now, most research on OWL 1.1 has been directed towards a new version of the description logics dialect of OWL 1.0[6], which is based on the $\mathcal{SROIQ}(\mathbf{D})$ description logic described by [Horrocks et al., 2006]. This DL extends $\mathcal{SHOIN}(\mathbf{D})$ with the following language constructs:

[5] See the WG website at `http://www.w3.org/2007/OWL/`.

[6] To avoid confusion, we will refer to the original OWL standard as OWL 1.0.

- Additional *property types*, these are:
  - *reflexive* and *irreflexive* properties, to express e.g. that every value is equal to itself, or that no effect can be its own cause, and
  - *symmetric* and *asymmetric* properties, that can express e.g. that every person is the relative of his own relatives, or that if something is bigger than something else, this does not hold vice versa.
- *Disjoint properties* allow to express e.g. the disjointness of the brotherhood and sisterhood properties: if Mary is Bob's sister, she is not his brother.
- *Local reflexivity* of properties enables us to express e.g. a narcissist (every narcissist likes himself). This takes the form of a 'Self' restriction on an object property.
- The feature of *property chain inclusions* – also called complex role inclusion – allows the 'inheritance' of some property over another property. For instance, the fact that is someone owns a car, he is also the owner of the car's parts.
- *Qualified cardinality restrictions* (QCR's) add the ability to restrict the cardinality of a property to a particular range, e.g. a table has exactly four legs as parts, but can have more parts which are not legs.
- It allows *punning* of individuals, classes and properties. This means that names can be treated as *any or all of* these types, e.g. the meaning of a name as a class is in no way affected by the meaning of a name as an individual. Punning enables convenient meta modeling without the drawbacks of RDF or OWL Full modeling.

Moreover:

- OWL 1.1 DL extends OWL 1.0 with *axiom annotations*, that is any axiom (such as a restriction) can be annotated. Furthermore, ways are currently being investigated to allow *'semantic' annotations* which can function as an extension to OWL semantics (similar to procedural attachments).
- It also provides a more elaborate scheme for *user defined datatypes*, such as restricted integer ranges, which is based on XML Schema datatypes[7].
- The last addition is syntactic sugar for some already expressible OWL 1.0 constructs: disjoint union, and negative property assertions.

The OWL 1.1 specification includes a number of so-called *fragments*, some of these are well-known tractable subsets of the OWL 1.1 DL specification, others are more expressive, but without the full semantics of OWL 1.1 Full. The motivation for providing these fragments is that many existing ontologies tend to use only a particular subset of the language constructs available in DL. Consequently, a significant increase of reasoner performance can be achieved through reasoning only

[7] See `http://www.w3.org/2007/OWL/wiki/Overview#ref-xml-schema-datatypes/`.

on relevant parts of the language. It is thought that a standard library of logical fragments with a particularly likeable trade-off between expressiveness and computational complexity can overcome some of the problems experienced when defining the OWL 1.0 Lite version. In particular the fragments:

- are expressive subsets of OWL 1.1, restricted by syntax. The semantics is provided by the OWL 1.1 DL specification.
- are logics that can handle at least some interesting inference services in polynomial time with respect to either:
  - the number of facts in the ontology, or
  - the size of the ontology as a whole.

Because of its early and wide adoption by both implementers and users, the OWL 1.1 effort has a good chance of reaching W3C recommendation status. A large part of the development of OWL 1.1's features has taken place during, and in between OWLED workshops, i.e. outside of the W3C standardization process[8]. At the start of the working group, efficient implementations of OWL 1.1 were already available in standard reasoners such as Pellet, FaCT++, Racer and KAON2[9].

To sum up, OWL 1.1 extends the OWL 1.0 standard with a number of relatively small, but significant improvements, and has a modular and extensible architecture in the form of language fragments, user defined datatypes and semantic annotations. Since OWL 1.1 preserves full backward compatibility with its predecessor, readjusting the terminological layer of LKIF, once OWL 1.1 becomes an official W3C recommendation, should not introduce any difficulties.

[8] OWLED: "OWL: Experiences and Directions". See `http://www.webont.org/owled/`.

[9] See `http://pellet.owldl.com`, `http://owl.man.ac.uk/factplusplus/`, `http://www.racer-systems.com` and `http://kaon2.semanticweb.org/` for respective descriptions and downloads.

5 Rule Layer

The rule layer of LKIF is supported by LKIF rules, a partly novel formalism developed under the ESTRELLA project. LKIF rules are not fully compatible with W3C standards, however none of the recommended rule languages, predominantly SWRL, seemed to satisfy the high expressiveness requirements derived from the goals of the project. At the time of writing of this summary, a new W3C proposal — RIF — has been put forward. This formalism will be shortly addressed at the end of this section.

5.1 LKIF Rules

LKIF rules are defeasible rules that amongst other suggest arguments to be made, and policies to be followed. It is used to express "rules" that are non-terminological and (therefore) defeasible in character. The LKIF rules extension is both an expressive extension of the LKIF logical language and a specific epistemological terminology, defining concepts such as *exception*, *assumption*[10].

5.1.1 Syntax

In this section the abstract s-expression syntax of LKIF rules is presented. As all parts of LKIF, rules can also be expressed in XML syntax, which, however, will be omitted in this summary.

Two basic constructs in the rule language are variables and symbols:

```
variable ::= symbol
constant-symbol ::= symbol
```

e.g. variables: `?x, ?person`; constant symbols: `agreement, lkif:Event`.

The two types are disjunct. Variables begin with a question mark character. Constant symbols may include a prefix denoting a namespace. A term is either a constant or a compound term:

```
constant ::= variable | constant-symbol
           | string | number | boolean

term ::= constant | '`' term | '(' constant-symbol term* ')'
```

e.g. terms: `?x, contract-1, `(red green)`.

Literals are atomic sentences or negations of atomic sentences. Since a constant-symbol can be used as an atom, LKIF rules provide a convenient syntax for a kind of propositional logic.

[10] The presentation of LKIF rules follows [Gordon, 2007], where some refinements with respect to the original D1.1 are introduced.

```
atom ::= constant-symbol | '(' constant-symbol term* ')'
literal ::= atom | '(' 'not' atom ')'
```

e.g. literals: `liable`, `(initiates event1 (possesses ?p ?o))`, `(not (children Tom '(Sally Susan)))`.

Finally, we define the syntax of rules as follows:

```
condition ::= literal
            | '(' 'unless' literal ')'
            | '(' 'assuming' literal ')'

conjunction ::= condition | '(' 'and' condition condition+ ')'

disjunction ::= '(' 'or' conjunction conjunction+ ')'

body ::= condition | conjunction | disjunction

head ::= literal | '(' 'and' literal literal+ ')'

rule ::= '(' 'rule' constant-symbol '(' 'if' body head ')' ')'
       | '(' 'rule' constant-symbol literal literal* ')'
```

The second rule form serves for representing rules with empty bodies, which could be interpreted as defeasible facts.

5.1.2 Semantics

An LKIF rule denotes a set of argumentation schemes, one for each conclusion of the rule, all of which are subclasses of a generic argumentation scheme for arguments from defeasible rules. Applying an LKIF rule is thus a matter of instantiating one of these argumentation schemes to produce a particular argument. Reasoning with LKIF rules is a process of applying these schemes to produce arguments to put forward in dialogues.

Definition 8 (Premises) *There are four kinds of argument premises:*

1. *If s is a sentence, then s is a* positive premise.
2. *If s is a sentence, then $\neg s$ is a* negative premise.
3. *If s is a sentence, then $\bullet s$ is an* assumption.
4. *If s is a sentence, then $\circ s$ is an* exception.

Definition 9 (Premises of Conditions) *Let p be a function mapping conditions of LKIF rules to argument premises, defined as follows:*

$$p(c) = \begin{cases} c & \text{if } c \text{ is an atomic sentence} \\ \neg s & \text{if } c \text{ is (not } s\text{)} \\ \bullet s & \text{if } c \text{ is (assuming } s\text{)} \\ \circ s & \text{if } c \text{ is (unless } s\text{)} \end{cases}$$

Arguments are of two kinds, *pro* arguments and *con* arguments. Argumentation schemes will be denoted as (presumptive) inference rules, by prefixing the conclusion of the inference rule with the direction of the argument. For this purpose, the following helping function maps the conclusions of LKIF rules to the appropriate argument conclusion. If a conclusion of an LKIF rule is an atomic sentence, s, then the rule is mapped to an argumentation scheme *pro* s. If a conclusion of the rule is a negated sentence, `(not` s`)`, then the rule is mapped to an argumentation scheme *con* s.

Definition 10 (Direction of Argument Conclusions) *Let d be a function mapping a conclusion of an LKIF rule to an atomic sentence prefixed by the direction of an argumentation scheme, as follows:*

$$d(c) = \begin{cases} \textit{pro } c & \textit{if } c \textit{ is an atomic sentence} \\ \textit{con } s & \textit{if } c \textit{ is } \texttt{(not } s\texttt{)} \end{cases}$$

The semantics of LKIF rules is defined by a mapping from rules to sets of argumentation schemes.

Definition 11 (Schemes for Arguments from LKIF Rules) *Let r be an LKIF rule, with conditions $a_1 \ldots a_n$ and conclusions $c_1 \ldots c_n$. Two premises, implicit in each LKIF rule, are made explicit here. The first, $\bullet v$, where $v =$* `(valid` r`)`*, makes the assumption that r is a* valid *legal rule explicit. The second, $\circ e$, where $e =$* `(excluded` r c_i`)`*, expresses the exception that r is* excluded *with respect to c_i.*

For each c_i in $c_1 \ldots c_n$ of r, r denotes the following argumentation scheme:

$$\frac{p(a_1) \ldots p(a_n), \bullet v, \circ e}{d(c_i)}$$

An argument is constructed by instantiating all variables in the argumentation scheme. The relations *valid* and *excluded* should be defined in models of particular legal domains. Finally, a rule is said to be *applicable* to a literal P if there exists a *pro* argumentation scheme, in case P is atomic, or a *con* argumentation scheme, if P is a negated atom, such that the conclusion of the scheme is unifiable with P, and all the premises are satisfied within the instantiation.

As mentioned before, a special module of LKIF-Core ontology has been specified for enabling the possibility of modeling LKIF rules as OWL objects and imposing the argumentation-theoretic semantics on their components. The module contains such concepts and properties as: `Argumentation`, `Rule`, `Premise`, `Antecedent`, `Consequent`, `prior`, `valid`, `unless`, etc.

5.1.3 Examples

The following are some examples of well-formed rules. Notice that the first two rules represent typical condition formulae, whereas the remaining two are rules with empty bodies.

```
(rule §-9-306-1
    (if (and (goods ?s ?c)
             (consideration ?s ?p)
             (collateral ?si ?c)
             (collateral ?si ?p)
             (holds (perfected ?si ?c) ?e)
             (unless (applies §-9-306-3-2 (perfected ?si ?p))))
        (holds (perfected ?si ?c) ?e)))

(rule §-9-306-2a
    (if (and (goods ?t ?c)
             (collateral ?s ?c))
        (not (terminates ?t (security-interest ?s)))))

(rule R1 (not (terminates T1 (security-interest S1))))
(rule R2 (collateral S1 C1))
```

5.2 Rule Interchange Format

The Rule Interchange Format (RIF) is a new proposal of W3C for an interchange format between rule languages. The formalism has not been fully developed yet and many issues are still under discussion of the RIF Working Group[11]. It is reasonable to expect, however, that at certain stage RIF will become a new W3C standard. Hence, the problem of compatibility of the LKIF rule layer with the Semantic Web should be probably reconsidered in the future.

The architecture of RIF is concentrated around the concept of a *dialect*, which denotes a language with a well-defined syntax and semantics. The Basic Logic Dialect of the Rule Interchange Format (RIF-BLD) is aimed to establish a common semantic denominator for all RIF languages based on logic, by referring to the semantics of FOL. Syntactically RIF-BLD should be compliant with the web technology, which means it has to adopt such features as resource-oriented vocabulary and compatibility with the XML-based standards. Nevertheless, it is intended that various RIF dialects will provide syntactic and semantic extensions for supporting different types of rules, also those not directly associated with web applications, such as: production rules, logic programming, FOL-based rules, reactive rules, and normative rules (integrity constraints).

Recently, the presentation syntax (EBNF) for RIF-BLD and its XML serialization have been introduced. Some interesting features of that specification are:

1. presence of signature statements, which provide convenient means for expressing meta-level properties of a rule language, such as syntactic constraints on well-formed terms and formulae, or definitions of different parts of vocabulary in extended dialects,

2. direct support for quantifiers and equality relation, which warrant high expressiveness of RIF,

[11] For a detailed overview see `http://www.w3.org/2005/rules/wiki/RIF_Working_Group`

3. syntactic freedom of constructing higher-order expressions.

Furthermore, a standard model-theoretic and proof-theoretic semantics have been defined for basic structures of RIF-BLD.

Altogether, the proposal seems very promising and is clearly under an intensive development. However, to be seriously considered as a functional interchange format for rule languages, RIF still requires some crucial improvements. Most important issues that are currently addressed by the RIF Working Group are:

1. obtaining compatibility with RDF[12],
2. specifying some useful extensions, for instance, for supporting negation (both classical and negation as failure)[13],
3. defining RIF Rulesystem Arrangement Framework (RIFRAF), a classification framework for rulesystems, fixing a unified vocabulary and a common set of discriminators for describing rule languages. RIFRAF will serve as a reference basis for the design of RIF dialects[14].

To summarize, RIF has good chances to become a new W3C standard as a rule interchange formalism. Having departed from SWRL, as insufficiently expressive for representing legal rules, LKIF might still maintain full compatibility with the Semantic Web technology by adjusting the specification of its rule layer to the requirements of RIF. In any case it does not mean that LKIF rules will have to be dropped for the sake of yet another formalism. Provided that suitable extensions for RIF are developed, LKIF will be equipped with appropriate translators allowing interchange between LKIF rule syntax and the respective RIF-compliant formats.

[12] See `http://www.w3.org/TR/rif-rdf-owl/`.
[13] See `http://www.w3.org/2005/rules/wg/wiki/A.2_Extension:_Negative_Conditions`.
[14] See `http://www.w3.org/2005/rules/wg/wiki/Rulesystem_Arrangement_Framework`.

6 Normative Statements

Deontic reasoning is inherent to the domain of law. LKIF supports the representation of deontic aspects of legal knowledge by providing the necessary terminology equipped with the minimal semantics required for modeling normative statements. As mentioned previously, LKIF takes a cautious approach on this layer of representation and does not commit itself to any particular deontic logic. Instead, only standard structural elements for expressing norms are specified. It is intended that the underlying semantics can be augmented in particular application scenarios in order to obtain more restrictive deontic settings.

Consequently, it is not possible to derive violations of norms automatically, at least not in the default setting defined in LKIF. Normative statements are interpreted rather as giving constraints on what can be considered a violation of the statement. The act of ascribing a normative qualification to something is essentially abductive.

6.1 Deontic Concepts

All elements of the deontic vocabulary, including the operators, are represented as OWL properties and classes in the module *Norm* of the LKIF-Core ontology.

Central to deontic reasoning is the notion of deontic choice (cf. [Hansson, 2001]), which states that if *it ought to be* α *given* β and an agent has the choice between bringing about $\alpha \sqcap \beta$ and $\neg\alpha \sqcap \beta$ then the agent should choose the former. In terms of a menu of alternatives this means that if $\mid \alpha \sqcap \beta \mid$ and $\mid \neg\alpha \sqcap \beta \mid$ are subsets of the agent's choice menu and $A_1 \in \mid \alpha \sqcap \beta \mid$ and $A_2 \in \mid \neg\alpha \sqcap \beta \mid$ then $A_1 \succ A_2$. The ordering relation $\succ$ captures here one of the most fundamental axiological intuitions, namely that of the *normative preference* or *subjunctive betterness* [Dayton, 1981]. The expression $A_1 \succ A_2$ is supposed to be interpreted in such context as A_1 *is normatively better than* A_2.

The deontic framework incorporated in LKIF builds on the notion of subjunctive betterness and accounts for its several formal derivatives: normatively_comparable, normatively_strictly_better, normatively_equivalent_or_worse, strictly_equivalent, etc. The table below lists some of the LKIF axioms that represent common-sense dependencies holding between the relations.

$$
\begin{aligned}
\mathsf{normatively\_strictly\_better} &\equiv \mathsf{normatively\_strictly\_worse}^{-1} \\
\mathsf{normatively\_equivalent\_or\_worse} &\equiv \mathsf{normatively\_equivalent\_or\_better}^{-1} \\
\mathsf{strictly\_equivalent} &\equiv \mathsf{strictly\_equivalent}^{-1} \\
\mathsf{normatively\_not\_equivalent} &\equiv \mathsf{normatively\_not\_equivalent}^{-1} \\
\mathsf{normatively\_equivalent\_or\_better} &\sqsubseteq \mathsf{normatively\_comparable} \\
\mathsf{strictly\_equivalent} &\sqsubseteq \mathsf{normatively\_equivalent\_or\_better} \\
\mathsf{strictly\_equivalent} &\sqsubseteq \mathsf{normatively\_equivalent\_or\_worse} \\
\mathsf{normatively\_not\_equivalent} &\sqsubseteq \mathsf{normatively\_comparable}
\end{aligned}
$$

Furthermore, domains and ranges of the relations are restricted only to the scope of deontic concepts. For instance:

$$\top \sqsubseteq \forall \mathsf{normatively\_strictly\_better}.\textsc{Obliged}$$

$$\top \sqsubseteq \forall \mathsf{normatively\_comparable}.\textsc{Normatively\_Qualified}$$
$$\top \sqsubseteq \forall \mathsf{normatively\_strictly\_worse}.\textsc{Disallowed}$$
$$\top \sqsubseteq \forall \mathsf{normatively\_equivalent\_or\_better}.\textsc{Allowed}$$
$$\top \sqsubseteq \forall \mathsf{strictly\_equivalent}.\textsc{Allowed}$$

In addition to subjunctive betterness LKIF introduces a selection of essential deontic concepts. The deontic operators (Normatively_Qualified, Allowed, Obliged, Disallowed) are given two kind of explications: an axiological one, expressed in terms of normative ordering relations introduced above, and an analytical — fixed via reference to other concepts (such as Permission, Obligation, etc.) The following four axioms formalize the first view on the deontic operators.

$$\textsc{Normatively\_Qualified} \equiv \exists \mathsf{normatively\_comparable}.\textsc{Normatively\_Qualified}$$
$$\textsc{Allowed} \equiv \exists \mathsf{normatively\_equivalent\_or\_worse}.\textsc{Normatively\_Qualified}$$
$$\textsc{Obliged} \equiv \exists \mathsf{normatively\_strictly\_worse}.\textsc{Disallowed}$$
$$\textsc{Disallowed} \equiv \exists \mathsf{normatively\_strictly\_better}.\textsc{Allowed}$$

The second interpretation, along with the meanings of the remaining concepts, is specified by the following definitions[15]:

$$\textsc{Norm} \equiv \exists \mathsf{qualifies}.\textsc{Normatively\_Qualified}$$
$$\textsc{Norm} \sqsubseteq \textsc{Qualification}$$
$$\textsc{Permission} \sqsubseteq \textsc{Norm}$$
$$\textsc{Permission} \equiv \exists \mathsf{allows}.\textsc{Allowed} \sqcap \forall \mathsf{allows}.\textsc{Allowed}$$
$$\textsc{Prohibition} \sqsubseteq \textsc{Permission}$$
$$\textsc{Prohibition} \equiv \forall \mathsf{allows}.\textsc{Obliged} \sqcap \exists \mathsf{allows}.\textsc{Obliged} \sqcap \forall \mathsf{disallows}.\textsc{Disallowed} \sqcap \exists \mathsf{disallows}.\textsc{Disallowed}$$
$$\textsc{Obligation} \equiv \textsc{Prohibition}$$
$$\textsc{Normatively\_Qualified} \equiv \exists \mathsf{qualified\_by}.\textsc{Norm}$$
$$\textsc{Allowed} \sqsubseteq \textsc{Normatively\_Qualified}$$
$$\textsc{Allowed} \equiv \exists \mathsf{allowed\_by}.\textsc{Permission}$$
$$\textsc{Obliged} \sqsubseteq \textsc{Allowed}$$
$$\textsc{Disallowed} \sqsubseteq \textsc{Normatively\_Qualified}$$
$$\textsc{Disallowed} \equiv \exists \mathsf{disallowed\_by}.\textsc{Prohibition}$$
$$\textsc{Strictly\_Allowed} \sqsubseteq \textsc{Allowed}$$
$$\textsc{Strictly\_Disallowed} \sqsubseteq \textsc{Disallowed}$$
$$\textsc{Allowed\_And\_Disallowed} \equiv \textsc{Disallowed} \sqcap \textsc{Allowed}$$
$$\mathsf{disallows} \sqsubseteq \mathsf{qualifies}$$
$$\mathsf{allows} \sqsubseteq \mathsf{qualifies}$$

Notice, that Allowed and Disallowed are not treated as disjoint classes. On the contrary, LKIF introduces the concept Allowed_And_Disallowed, thus explicitly permitting normative conflicts to occur in the representation without entailing inconsistency. The design is intended for accommodating exceptions, the so-called contrary-to-duty situations, or coexistence of different normative theories within one model, whose conflicting provisions are to be adjudicated by higher level rules.

[15] For a complete listing of the axioms in the module *Norm* see Deliverable 1.4 [Breuker et al., 2007].

6.2 Semantics

In this section we briefly outline the mapping of normative statements to OWL-DL. The deontic constructs that are of interest here are dyadic obligations, prohibitions, and permissions of the form $O(\alpha \mid \beta)$, intuitively meaning that α *is obliged if* β *is true, according to the normative theory under consideration.* For simplicity we address only statements based on modal frames representing a unique normative theory $M = \langle W, \preceq, V \rangle$, where W is a set of worlds, $\preceq$ is the accessibility (normative preference) relation constituting the represented theory, and V is a valuation function over propositions. The extension to multiple theories is straightforward, though requires employing separate OWL properties for respective preference relations.

OWL-DL can be understood as a subset of labeled modal logic (cf. [Horrocks et al., 2003, Haarslev and Moller, 2001]). Taking such a perspective individuals are interpreted as possible worlds, whereas their types as propositions true in those worlds. As a result we translate an obligation $O(\alpha \mid \beta)$, or a prohibition $F(\neg\alpha \mid \beta)$ to a pair of DL axioms:

$$(\beta \sqcap \alpha) \sqsubseteq \neg\exists \preceq (\beta \sqcap \neg\alpha) \qquad (\beta \sqcap \neg\alpha) \sqsubseteq \exists \preceq (\beta \sqcap \alpha)$$

This says that in a world where $(\beta \sqcap \alpha)$ is true, there is no world normatively equal or better accessible where $(\beta \sqcap \neg\alpha)$ is true, and in a world where $(\beta \sqcap \neg\alpha)$ is true, there is some world equal or better accessible where $(\beta \sqcap \alpha)$ is true, according to the theory. Due to syntactic restrictions of OWL-DL the representation of the axioms requires the use of some auxiliary concepts. Thus an obligation or the corresponding prohibition should be in fact formalized as:

$$C_1 \equiv \beta \sqcap \alpha \qquad C_2 \equiv \beta \sqcap \neg\alpha \qquad C_1 \sqsubseteq \neg\exists \preceq C_2 \qquad C_2 \sqsubseteq \exists \preceq C_1$$

Where C_1, C_2 and C_3, are new concept names. Analogically we translate permission $P(\alpha \mid \beta)$ to:

$$(\beta \sqcap \neg\alpha) \sqsubseteq \exists \preceq (\beta \sqcap \alpha)$$

meaning that in a world where $(\beta \sqcap \neg\alpha)$ is true, there is an access to some normatively equal or better world where $(\beta \sqcap \alpha)$ is true. Further, we give it the following OWL-DL representation:

$$C_1 \equiv \beta \sqcap \alpha \qquad C_2 \equiv \beta \sqcap \neg\alpha \qquad C_2 \sqsubseteq \exists \preceq C_1$$

What follows from the above presentation is that all normative statements appropriately represented in OWL automatically obtain an additional interpretation based on Kripke frame semantics, which underpin typical deontic logics. Consequently, it can be proven that some common-sense properties hold for the default deontic framework defined in LKIF:

1. Some worlds are normatively incomparable.
2. What is obligatory is permitted: $O(\alpha \mid \beta) \sqsubseteq P(\alpha \mid \beta)$ is true.
3. The impossible is not obligatory: $\neg O(\neg\alpha \mid \alpha)$ is true.

4. There are no conflicting obligations within a single normative theory. The obligations $O(\alpha \mid \beta)$ and $O(\neg\alpha \mid \beta)$ are inconsistent: $\neg(O(\alpha \mid \beta) \sqcap O(\neg\alpha \mid \beta))$ is true.

5. The obligation $O(\alpha \mid \beta)$ and permission $P(\neg\alpha \mid \beta)$ are inconsistent: $\neg(O(\alpha \mid \beta) \sqcap P(\neg\alpha \mid \beta))$ is true.

6. The sentences $O(\alpha \mid \top)$, $O(\beta \mid \alpha)$, $O(\neg\beta \mid \neg\alpha)$, representing a contrary-to-duty situations, are only satisfied by the ordering identified in [Chisholm, 1963], i.e. $\mid \neg\alpha \sqcap \beta \mid\prec\mid \neg\alpha \sqcap \neg\beta \mid\prec\mid \alpha \sqcap \neg\beta \mid\prec\mid \alpha \sqcap \beta \mid$.

6.3 Examples

For an illustration consider a simplified formalization of one of the articles from the Dutch traffic code RVV-90.

> Article 12: It is not permitted to overtake a vehicle directly before or on a pedestrian crossing.

$$
\begin{aligned}
\textsc{Behavior}_1 &\equiv \textsc{Overtaking\_Vehicle} \sqcap \\
&\quad \exists \mathsf{directly\_before\_or\_on}.\textsc{Pedestrian\_Crossing} \\
\textsc{Behavior}_2 &\equiv \neg\textsc{Overtaking\_Vehicle} \sqcap \\
&\quad \exists \mathsf{directly\_before\_or\_on}.\textsc{Pedestrian\_Crossing} \\
\textsc{Behavior}_1 &\sqsubseteq \exists \mathsf{normatively\_strictly\_better}.\textsc{Behavior}_2 \\
\textsc{Behavior}_2 &\sqsubseteq \neg\exists \mathsf{normatively\_equivalent\_or\_better}.\textsc{Behavior}_1
\end{aligned}
$$

The representation allows for interpreting the statement as a proper dyadic obligation. Observe, that if we augment the last axiom with the additional restriction $\exists \mathsf{normatively\_strictly\_worse}.\textsc{Behavior}_1$ then it will hold:

$$\textsc{Behavior}_2 \sqsubseteq \textsc{Obliged} \Leftrightarrow \textsc{Behavior}_1 \sqsubseteq \textsc{Disallowed}$$

which is clearly the desired classification. The strengthening is required here due to certain limitations of OWL's expressiveness.

Alternatively, one may use deontic concepts in rules, for instance:

```
(rule §-12
    (if (and (Overtaking_Vehicle ?x)
             (Pedestrian_Crossing ?y)
             (directly_before_or_on ?x ?y))
        (Disallowed ?x)))
```

In such an approach, the modal interpretation of the statement remains somewhat implicit and can be only partially retrieved from the definition of the concept Disallowed. Still, the knowledge about deontic consequences of the statement is directly accessible in the model.

7 Conclusions

In this document we have summarized the content of the Deliverable 1.1 for the ESTRELLA project, reporting on the formal specification of the Legal Knowledge Interchange Format.

Concluding, LKIF is a compound formalism based on a well-defined semantics and syntax. At the current stage of development it encompasses Web Ontology Language in the DL variant, meant predominantly for expressing terminological knowledge, and the language of LKIF rules, where rules can be in principle translated and incorporated into OWL knowledge bases using the framework defined in the *Argumentation* module of the LKIF-Core ontology. Another module, called *Norm*, has been designed for supporting OWL representation of normative statements.

On the three layers of expressiveness LKIF is provided with three different types of semantics. The standard model-theoretic semantics underlies terminological knowledge represented in OWL-DL. Rules are interpreted on the grounds of argumentation theory, where rule constructs are mapped to corresponding parts of argumentation schemes. Finally, well-formed normative statements can be associated with certain deontic formulae and thus analyzed in terms of the Kripke semantics. Consequently, there are also several abstract and pretty-printing syntaxes that facilitate presentation of particular types of knowledge expressible in LKIF. The common denominator and the basic concrete syntax for the whole LKIF is XML.

The design of LKIF has been driven by two major objectives: complying to Semantic Web standards and gearing the expressiveness of the language to the requirements of the legal domain.

The first goal has been achieved to the extent to which OWL-DL has been employed as a representation formalism. Compatibility of the non-standard rule layer remains at the moment an open issue. A possible solution to the problem might be found in the nearby future through the Rule Interchange Format — a new W3C proposal — provided that suitable RIF dialects are developed.

The legal-oriented expressiveness is in general exhibited in the choice of types of knowledge that obtain direct support in LKIF and, more specifically, in the selection of expressive means available on every layer. Terminological layer is supplied with the LKIF-Core ontology of basic legal concepts, rule formalism — with operators and semantics facilitating construction of legal arguments, and the layer of normative statements — with the necessary vocabulary and intended semantics for representing deontic content of legal norms.

References

[Baader et. al, 2003] Baader, F., McGuinness, D. L., and Nardi, D., editors (2003). *Description Logic Handbook*. Cambridge University Press.

[Bench-Capon and Coenen, 1992] Bench-Capon, T. and Coenen, F. P. (1992). Isomorphism and legal knowledge based systems. *Artificial Intelligence and Law*, 1(1):65-86.

[Boer et al., 2007] Di Bello, M., van den Berg K., Boer, A., Förhécz, A., Gordon, T., Vas, R. (2007), *Deliverable 1.1, Specification of the Legal Knowledge Interchange Format*, Technical Report, ESTRELLA Project.

[Breuker et al., 2007] Breuker, J., Hoekstra, R., Boer, A. editors, van den Berg, K., Sartor, G., Rubino, R., Wyner, A., Bench-Capon, T., (2007), D1.4 - *OWL Ontology of Basic Legal Concepts (LKIF-Core)*, Technical Report, ESTRELLA Project.

[Chisholm, 1963] Chisholm, R. M. (1963). Contrary-to-duty Imperatives and Deontic Logic, *Analysis*, 24:33-36.

[Dayton, 1981] Dayton, E., (1981). Two Approaches to Deontic Logic, *The Journal of Value Inquiry*, 15:137-147

[Di Bello and van den Berg, 2007] Di Bello, M., van den Berg, K., *Evaluating the RMG Language for Modeling Legislation*.

[ESTRELLA, 2006] ESTRELLA (2006). ESTRELLA Technical Annex (IST-4-027655). Technical report, European Commission.

[Gordon, 2007] Gordon, T. F., (2007). Constructing Legal Arguments with Rules in the Legal Knowledge Interchange Format (LKIF), forthcoming.

[Haarslev and Moller, 2001] Haarslev, V. and Moller, R. (2001). Description of the RACER System and Its Applications. In *Proc. International Workshop on Description Logics* (DL- 2001), pages 131141.

[Hansson, 2001] Hansson, S. O. (2001). *The Structure of Values and Norms*. Cambridge University Press, UK.

[Horrocks et al., 2003] Horrocks, I., Patel-Schneider, P., and van Harmelen, F. (2003). From $\mathcal{SHIQ}$ and RDF to OWL: The Making of a Web Ontology Language. *Journal of Web Semantics*, 1(1):726.

[Horrocks et al., 2006] Horrocks, I., Kutz, O., Sattler, U., (2006). The Even More Irresistible $\mathcal{SROIQ}$, *Proceedings of the Tenth International Conference on Principles of Knowledge Representation and Reasoning*, 57-67

[Rissland et al., 2003] Rissland, E. L., Ashley, K. D., and Loui, R. P. (2003). AI and Law: A Fruitful Synergy. *Artificial Intelligence*, 150(12):115.

[van den Berg et al., 2006] van den Berg, K., Gordon, T. F., Kordelaar, P., Lee, J., Sekat, S., and Wyner, A. (2006). D1.2 *Formal Specifications of the Knowledge Formats of the Participating LKBS Vendors*. Technical Report, ESTRELLA Project.