An OWL ontology library representing judicial interpretations

Editors: Pompeu Casanovas, Universitat Autònoma de Barcelona, Spain, and Victoria University, Australia; Monica Palmirani, University of Bologna - CIRSFID, Italy; Silvio Peroni, University of Bologna, Italy, and National Research Council, Italy; Tom van Engers, University of Amsterdam, The Netherlands; Fabio Vitali, University of Bologna, Italy
Solicited reviews: Andrew Koster, Federal University of Rio Grande Do Sul, Brazil; Thomas Bruce, Cornell University, USA; Nick Bassiliades, Aristotle University of Thessaloniki, Greece

Marcello Ceci[a*] and Aldo Gangemi[b,c]
[a] *GRCTC, University College Cork, College road, Cork, Ireland*
[b] *ISTC-CNR, Via S. Martino della Battaglia 44, 00185 Rome, Italy*
[c] *LIPN, Université Paris 13, Sorbonne-Cité-CNRS, Paris, France*

Abstract. The article introduces JudO, an OWL2 ontology library of legal knowledge that relies on the metadata contained in judicial documents. JudO represents the interpretations performed by a judge while conducting legal reasoning towards the adjudication of a case. To the aim of this application, judicial interpretation is intended in the restricted sense of the acts of judicial subsumption performed by the judge when he considers a material instance (*token* in Searle's terminology), and assigns it to an abstract category (*type*). The ontology library is based on a theoretical model and on some specific patterns that exploit some new features introduced by OWL2. JudO provides meaningful legal semantics, while retaining a strong connection to source documents (fragments of legal texts). The application task is to enable detection and modeling of jurisprudence-related information directly from the text, and to perform shallow reasoning on the resulting knowledge base. The ontology library is also supposed to support a defeasible rule set for legal argumentation on the groundings of judicial decisions.

Keywords: legal knowledge modeling, ontology design patterns, case-based legal reasoning, judicial interpretation, OWL2

[*] Corresponding author. E-mail: marcello.ceci@gmail.com

«I see, these books are probably law books, and it is an essential part of the justice dispensed here that you should be condemned not only in innocence but also in ignorance».

- Franz Kafka, *The Trial*.

1. Representing the Judicial Framework

Precedents (or *case law*) are core elements of legal knowledge worldwide: by settling conflicts and sanctioning illegal behaviors, judicial activity enforces law provisions within national borders, supporting the validity of laws as well as the sovereignty of the government that issued them. Moreover, precedents are a fundamental source for legal interpretation, to the point that the exercise of jurisdiction can influence the scope of the same norms it has to apply, both in common law and civil law systems – although to different extents.

Capturing the semantics of human-created texts to be processed by machines is not a linear process. In order to provide a comprehensive representation of the contents of a document it is necessary to adopt multiple perspectives, and to account for different aspects and granularity of representation. Legal documents require special attention when representing their semantics, as they do not typically express factual knowledge, rather codifying an order of an authority that can be translated by means of logical operators, but whose syntax is not fixed. Unlike a generic text, where the intended meaning of the combination of signs is either common knowledge or is explained by the author, interpretation of legal documents is a different matter. The language used is important by itself, its conventional meaning being codified by the legal system. However, it is also commonly accepted that assigning a meaning to legal dispositions is not straightforward: there are gray areas in the interpretation of legal, open-textured concepts, and the effects of legal acts are susceptible to change in time, either depending on a change of the legal text itself, or on external influences (i.e. other norms or judgments). The AI & Law research community has gathered significant results on this topic since the 1980s, with different approaches: legal case-based reasoning [2,11], ontology-based systems [34], and formal argumentation [24,26,44].

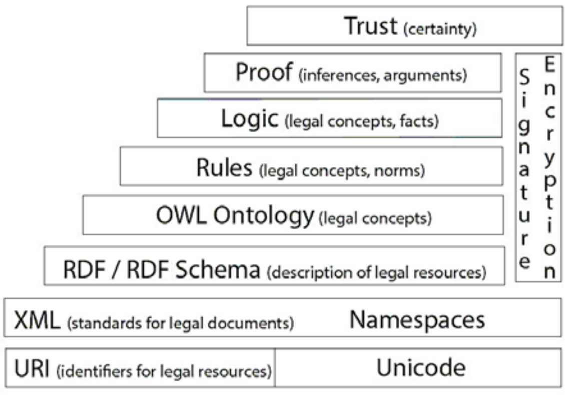

Trust (certainty)
Proof (inferences, arguments)
Logic (legal concepts, facts)
Rules (legal concepts, norms)
OWL Ontology (legal concepts)
RDF / RDF Schema (description of legal resources)
Signature
Encryption
XML (standards for legal documents)
Namespaces
URI (identifiers for legal resources)
Unicode

Fig. 1. Tim Berners Lee's Semantic Web layer cake, adapted to the legal domain in [47].

This papers covers part of a research (see [13]) whose aim is to define a Semantic Web framework for precedent modeling, by using knowledge extracted from text, metadata, and rules [5], while maintaining a strong text-to-knowledge morphism, in order to *fill the gap* between legal document and its semantics [38]. The input to the framework includes metadata associated with judicial concepts and an ontology library representing the structure of case law.

The research relies on previous efforts of the community in the field of legal knowledge representation [35] and rule interchange for applications in the legal domain [27]. The issue of implementing logics to represent judicial interpretation has already been faced e.g. in [9,22], albeit only in sample cases. The aim of the research is to apply legal theories to a set of real legal documents, defining OWL axioms in a *Judicial Ontology Library* (JudO) that provides a semantically expressive representation and a solid ground for a (future) legal argumentation system based on a defeasible subset of predicate logics. The JudO ontology library thus constitutes the cornerstone for semantic tools to enrich and reason on the XML mark-up of precedents (i.e. the metadata of case-law), supporting legal reasoning in the large.

Some new features in the recent version of OWL (OWL2, see [53]) unlock useful reasoning features for legal knowledge, especially if combined with defeasible rules. The main task is thus to formalize legal concepts and argumentation patterns contained in a judgement, with the following requirement: *to check, validate and reuse the discourse of a judge - and the argumentation he produces - as expressed by judicial text*. In order to achieve this, four different models that make use of standard languages from the Semantic Web layer cake (Figure 1) have been used:

a. A **document metadata structure**, modeling the main parts of a judgement, and creating a bridge between a text and its semantic annotations of legal concepts;
b. A **legal core ontology**, modeling abstract legal concepts and institutions contained in a rule of law [16];
c. A **legal domain ontology**, modeling the legal concepts involved in a specific domain of case-law;
d. An **argumentation system** [15], modeling the structure of argumentation (arguments, counterarguments, premises, conclusions, rebuttals, proof standards, argument schemes, etc.).

This work deals with issues related to the *core* and *domain ontologies* – (b.) and (c.) – which organize the metadata annotating the text of judicial decisions and infer relevant knowledge about precedents. The metadata structure is obtained from the Akoma Ntoso standard (see 3.1.), while multiple solutions are being tested for building argumentation out of the ontology library: an application of the ontology library to the Carneades Argumentation System is described in [15], while future research will focus on applications on Drools (see [42]) and SPINdle (see [43]).

The paper is structured as follows: section 2 presents the requirements and the design methods for the ontology library; section 3 describes the ontology library design, and how it is used to represent knowledge related to judicial interpretation. The method is exemplified in section 4 with reference to a sample of Italian case law. Section 5 presents an evaluation of the ontology, discussing related work in both legal ontology and legal reasoning fields, and some remaining issues with the proposed solution.

2. Tasks and applications

This research applies state-of-the-art techniques in ontology design and DL reasoning for the representation of knowledge extracted from legal documents, stressing OWL2 axiomatization capabilities in order to provide an expressive representation of judicial documents, and a solid ground for an argumentation system that uses a defeasible subset of predicate logics.

Modeling judicial knowledge involves the representation of situations where strict deductive logic is not sufficient to reproduce the legal reasoning as performed by a judge. In particular, defeasible logics [28] seem necessary to represent the legal rules underlying judicial reasoning. For example, many norms concerning contracts are not mandatory: they could be overruled by a different legal discipline through specific agreements between the parties. The problem of representing *defeasible* rules is a core problem in legal knowledge representation.

Moreover, argumentation theories (including the dialogue model of adjudication by [44], and argumentation schemes by [26]) introduce tools that are fundamental to perform effective reasoning on legal issues. This perspective adopts a procedural view on argumentation, which is necessary in order to properly represent those processes in an argument graph.

However, not all reasoning on judicial knowledge needs defeasible rules and argumentation, therefore we can safely apply classical deductive reasoning to a substantial subset. For example, the fact that most legal concepts do not admit both necessary and sufficient conditions is sometimes regarded as a limitation for a classical representation of legal concepts. However, it is common practice in domain ontologies to introduce mostly necessary conditions, which have a major role in reasoning, although enabling a smaller amount of inferences. In addition, some relevant domain concepts in law can be designed by class axioms instead of rules, so providing an explicit account of domain-level classical reasoning. The JudO class `Relevant_Ex<rulename>`, under which all instances relevant to a specific law are automatically classified (see 3.3.1.), is an example of such design choice.

The ontologies introduced in this paper address the classical subset of legal knowledge, in order to enrich the metadata annotating a legal document by performing deductive reasoning, and thus preparing a knowledge base for additional reasoning performed by tools based on deontic defeasible logics and argumentation schemes.

Following the requirement schema for legal ontologies that has been proposed in [19], the JudO ontology library is supposed to satisfy the following functional, domain, and application requirements. Functional requirements include:

- **Text-to-knowledge morphism**: the aim is to design the knowledge that can be extracted from a (textual) judicial decision, or a fragment of it, as a module in an ontology library, so that each module constitutes a particular morphism of the legal meaning expressed by that text [39]. This means that the ontology should be easily extended with entities extracted from the legal text, and it should contain only as many constraints as needed by judicial reasoning, without over-constraining it with unneeded axioms (i.e. uncertain sufficient conditions, unsure disjointness, etc.);

- **Distinction between document layers**: the ontologies must clearly distinguish between the legal text (the medium and expression layer), its meaning (the legal concepts and rules contained in the text), and the entities referred by the text. In principle, different (and even inconsistent) legal meanings can be expressed by the same legal text. Achieving distinction between document layers involves the identification of frames from different layers (see [20,22] for examples of layered, frame-oriented ontology design in law):
  - *Social frames*, concerning the effects of the legal text in the social world (extra-legal perspective);
  - *Procedural frames*, concerning the effects of the legal text in the identification of different steps in a legal proceeding;
  - *Substantial frames*, concerning the effects of the legal text in the application of the norms it expresses.
- **Shallow reasoning** on judicial knowledge: the ontologies must enable reasoning on material circumstances, legal concepts and judicial interpretations contained in precedents. In order to achieve this, JudO has to:
  - *Identify* the acts which have legal force, distinguishing them on the basis of their strength (this has been achieved, for example, by distinguishing between "weak links" created by contracts and "strong links" created by judicial interpretation, which can overrule previous ones);
  - *Create a conceptual frame* bound together by the acts with legal force. JudO is based on the notion of *qualifying legal expression* (see section 3.2.1), whose function is to create links between legal concepts under a same hat. The framing works by modeling those links as a relation between the qualification (the legal act) and the qualified elements[1]. In practice, these links do not contribute to uniquely characterizing a legal object (because several – and possibly inconsistent – qualifications may involve the same object), but rather constitute a net of relations that provide the bread and butter of judicial interpretation. In the legal domain, relations seem to be more important than categories[2].
- **Querying**: being able to perform complex querying, e.g. by using SPARQL-DL [51], on qualified parts of a judgement text. For example, performing queries that encode a question such as: "*retrieve all the judgements in the last year, with a dissenting opinion, in the e-commerce field, and where the main argument of the decision is the application of Consumer Law, art. 122*";
- **Supporting text summarization**: detecting relevant parts of a judicial text by reasoning on semantic annotations jointly with judicial ontologies;
- **Modularity**: JudO should define modules that axiomatize concepts common to as many domain ontologies as possible, which in turn should be automatically imported depending on the domain and task at hand;
- **Supporting case-based reasoning**: performing legal case-based reasoning by using the ontology reasoner in combination with a set of rules and a rule engine (see [15]). Frame-based judicial qualification is particularly appropriate to this requirement.

Judicial ontologies are intended to create an environment where knowledge extracted from the decision text can be processed and managed, and reasoning on the judicial interpretation that grounds the decision is enabled. Reasoning intends to satisfy the following domain requirements[3] (also known as *competency questions*, see [29]):

- **Finding relevant precedents** that are not explicitly cited in the decision. In order to

[1] This is in line with the Descriptions and Situations framework, as used in e.g. [20,22].

[2] For example, signing a contract clause at the end of the page it is contained in could be considered as a specific signing of the clause in a judgement A, while not so in a judgement B. With JudO, we do not intend to determine which interpretation is more accurate, but rather to annotate both of them, together with the contextual information about the different judgements.

[3] See [13] for an implementation of the ontology library into the Carneades Argumentation System.

achieve that, JudO should model entities such as:
  * laws cited;
  * legal figures evoked;
  * factors present in the material circumstances;
- **Validating adjudications** of a judge about the claims brought forward by the parties in a real legal case on the basis of applicable rules, accepted evidence, and interpretation. To perform that, the ontology needs to:
  * reproduce the semantics of legal consequences brought forward by legal rules;
  * be able to automatically infer its application.

  Such inference can then be compared to the outcome of the real legal case (classified in the ontology as an instance of the `Adjudication` class).
- **Suggesting legal rules, precedents, or circumstances** that might lead to a different adjudication of the claim. In order to achieve this:
  * the legal concepts $c_{1...n}$ applied by a judgement j ($a^{cj}{}_{1}$) must contain information (coming from other precedents) about their other known applications $a^{cj}{}_{2...m}$. In this way, once a legal concept c_i is evoked, we can compare each application $a^{cj}{}_{i}$ to other judgements, which could be inconsistent with $a^{cj}{}_{1}$;
  * the galaxy of connections between the pieces of knowledge in the ontology can be based on either crisp or fuzzy categories, since a main requirement is to emphasize indirect connections between concepts. Certainly, in order to take the most advantages from this assumption, we may need to add fuzzy reasoning to JudO OWL axioms (cf. [7,8]).

The structure of the ontology library also aims at integrating the representation of legal concepts at different layers of legal interpretation, as when considering concepts in laws together with concepts in legal principles.

Practical applications of the ontology library include:

- **Compliance checking** of contract drafts, using a word processor plugin that employs NLP techniques to recognize sentences and clauses that could be relevant under consumer law and then representing and reasoning with it;
- **Juridical analysis tools** for legal professionals, enriching case-law collections by semantically relating and grouping precedents for lawyers to browse, making the precedent extraction process for legal cases easier and more effective;
- **Judgement management tools** for courts and tribunals, useful to evaluate and optimize judgements (integrated into a word processor to assist judges while writing judgements, highlighting missing elements in the decision's groundings, which could then constitute grounds for appeals);
- **Impact analysis tools** for legislators, providing a list of (common or uncommon) judicial interpretations for a given law, in order to take them into account when modifying that law;
- **Tools representing formalized legal doctrine** and case law, where legal experts could rely on a social platform to share their views and interpretations on a law or a precedent, by using a graphical interface and a formal argumentation structure instead of plain text.

3. Ontology Design

The intended application of JudO is based on a multi-layer paradigm, where a legal resource is managed in separate, mutually connected, levels, which are organized in order to allow multiple annotation, interpretation, and classification, with representation redundancy. The annotation layer consists of the following elements:

- **Text annotation in XML**: the Akoma Ntoso standard [4,52] grants proper mark-up of the structure of judgements and citations;
- **Metadata annotation**: the Akoma Ntoso *metadata* block captures not only the metadata concerning the lifecycle of the document (e.g. workflow of the trial, formal steps, jurisdiction, level of judgements), but also legal qualification about relevant parts of the decision, such as a minority report or a dissenting opinion;
- **Ontology annotation**: external OWL definitions linked to the XML document are used;

- **Rules**: unfortunately OWL, even with the functionalities of version 2, is unable to represent complex and defeasible legal arguments. It is therefore necessary to extend the model with rule modeling for argumentation representation.

The JudO ontology is designed into two main modules (see also [16]):

- a **Core Ontology** describing the constituents of a precedent in terms of general concepts, designed as specification the Description and Situations framework as implemented in DOLCE-Ultralite+D&S [4], and aligned to the LKIF-Core legal ontology;[5]
- a **Domain Ontology** representing the concepts and rules expressed by the Italian *Codice del Consumo* (Consumer Code), by *artt.* (articles) 1241 and 1242 of the Italian Civil Code, as well as all relevant knowledge extracted from a set of Italian judgements containing interpretation of private agreements in the light of those laws.

Our design method is based on a middle-out methodology that incrementally integrates a bottom-up approach for capturing and modeling legal domain ontologies, and a top-down one for modeling core ontology classes and argumentation theory components. The middle-out methodology is implemented here by using pattern-based design [6,19], where ontology design patterns are extracted from judicial text, and, whenever possible, they are defined complying with the core ontology according to requirements.

The notion of *judicial interpretation* is the central one. It involves:

- **acts of interpretation**, which take into account facts, and apply legal rules (legal statuses) to it;
- **interpretations of a legal text**, since a same phrasing may give rise to alternative interpretation acts, depending on the meaning given to the words.

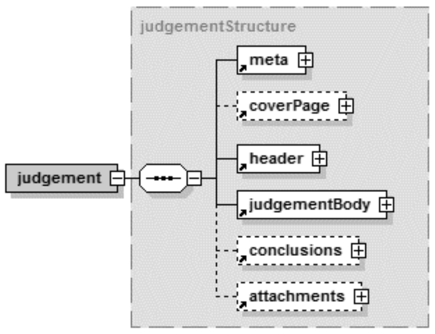


Fig. 2. Judgement structure in Akoma Ntoso.

The Description and Situations (D&S) ontology design pattern [22] [6] provides the relations to distinguish between *facts*, interpretive actions that construct *descriptions* to interpret those facts, and the possible resulting interpretations (*situations*). D&S (as axiomatized in [22]) also supports the link between interpretive actions, and the information objects (*expressions*) that are actual parts of the facts in the legal world that are taken into account by JudO.

The *qualification pattern* (see 3.2.1.) is aimed at capturing some aspects of legal interpretation, while keeping an open approach in order to maximize the results of the reasoning, since in the legal field even remote, apparently counterintuitive inferences may be important.

Evaluation has been performed on a sample set of Italian case law including 27 decisions of different grades (Tribunal, Court of Appeal, Cassation Court) concerning the legal field of oppressive clauses in Consumer Contracts. The matter is specifically disciplined in the Italian “Codice del Consumo” (Consumer Code), as well as in many non-Italian legal systems, so that an extension of this research to foreign decisions (and laws) can be envisaged.

Contract law is an interesting field because the (either automatic or manual) markup of contract parts allows highlighting single clauses and their comparison to general rules, as well as the case law concerning the matter. Contract markup can be used to perform semi-automatic compliance checking of a contract draft. The domain considered is also interesting for knowledge representation, because it involves situations where strictly deductive logic is not sufficient to represent the legal reasoning as

[4] http://www.ontologydesignpatterns.org/ont/dul/DUL.owl

[5] LKIF Core base URI is http://www.estrellaproject.org/lkif-core/lkif-core.owl, but it is not resolvable at the moment of finalizing this paper. You might use instead: https://github.com/RinkeHoekstra/lkif-core as a page for its source code.

[6] See also [23] for a previous version tailored to norm dynamics.

performed by a judge. In particular, defeasible logics [28] seems needed to represent the legal rules underlying judicial reasoning. For example, many norms concerning contracts are not mandatory: they could be overruled by a different legal discipline through specific agreements between the parties. The problem of representing defeasible rules, in fact, is a core problem in legal knowledge representation. Exploring how OWL2 could set the background for the application of defeasible logic is therefore one of the goals of the present research. See sections 4 and 5 for a presentation of the results achieved by the judicial framework.

3.1. Judgement Structure

Judgement in Akoma Ntoso [4] is a type of XML document modeled to detect the relevant parts of a text describing a precedent (Figure 2): a *header* for capturing the main information such as parties, court, neutral citation, document identification number; a *body* for representing the main part of the judgement, including the decision; a *conclusion* for including the signatures.

The *body* part is divided into four main blocks: *introduction*, where usually (especially in common law decisions) the story of a trial is introduced; *background*, dedicated to the description of the facts; *motivation*, where the judge introduces the arguments supporting his decision; *decision*, where the final outcome is given by the judge.

This partitioning allows the highlighting of facts and factors pertaining to the judgement: in the *motivation* part, arguments and counterarguments are detectable, while in the *decision* part lies the conclusion of the legal argumentation process. Those “qualified” fragments of text are annotated by legal experts with the help of a special editor (e.g. Norma-Editor, presented in [37]) that is handy to create links between text, metadata and ontology classes.

3.2. Core Ontology

The judicial core ontology[4] (Figure 3), or JudO, implements the *qualification ontology design pattern*. JudO defines OWL entities for the main concepts and relations in the judicial legal domain, dealing with *judicial decisions*. Core ontologies are typically domain-generic; however, being the legal domain too large and heterogeneous, several core ontologies can be designed. The ontology presented here is conceived to represent interactions in Civil Law, especially where contracts, laws and judicial decisions are concerned. For other domains, e.g. public contracts, administrative law, tort law, etc. adaptations are needed.

3.2.1. Qualifying Legal Expressions

[4] http://codexml.cirsfid.unibo.it/ontologies/judging_contracts_core.owl.

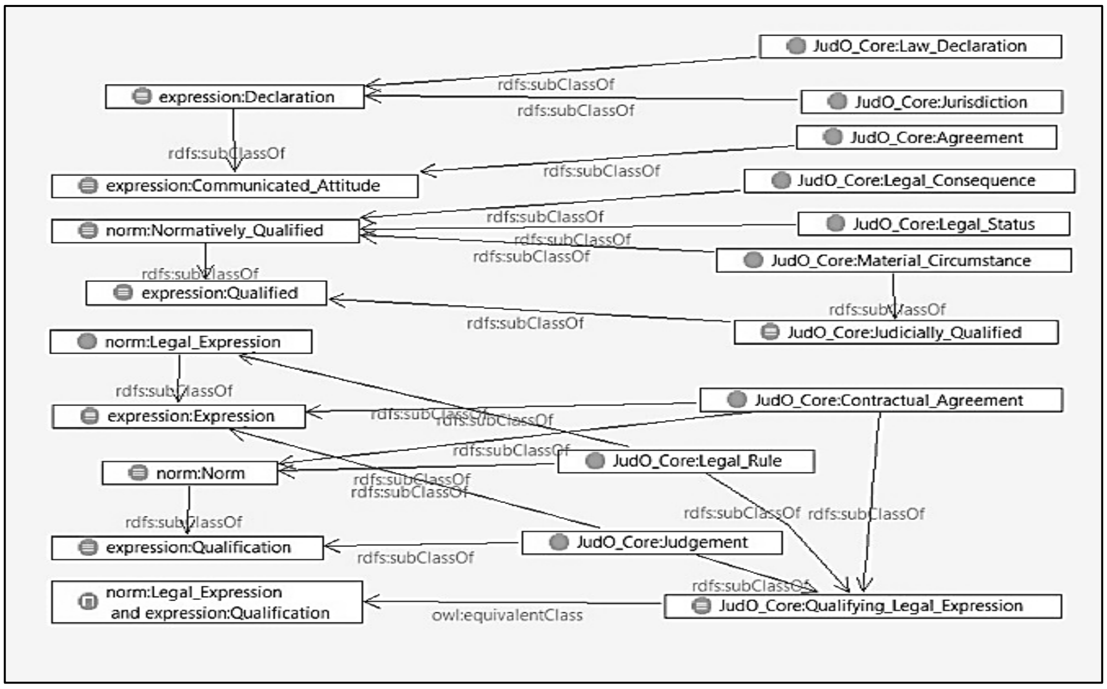

JudO_Core:Law_Declaration
JudO_Core:Jurisdiction
JudO_Core:Agreement
JudO_Core:Legal_Consequence
JudO_Core:Legal_Status
JudO_Core:Material_Circumstance
JudO_Core:Judicially_Qualified
JudO_Core:Contractual_Agreement
JudO_Core:Legal_Rule
JudO_Core:Judgement
JudO_Core:Qualifying_Legal_Expression
expression:Declaration
expression:Communicated_Attitude
norm:Normatively_Qualified
expression:Qualified
norm:Legal_Expression
expression:Expression
norm:Norm
expression:Qualification
norm:Legal_Expression and expression:Qualification
rdfs:subClassOf
owl:equivalentClass

Fig. 3. JudO Core Ontology (right) alignment to LKIF-Core (left).

The backbone of the JudO Core Ontology is constituted by three classes: `Qualifying_Legal_Expression`, `Qualification`, and `Qualified`.

In order to understand those classes and their main relations (Figure 4), we need to get an intuition (here described in non-strictly legal terms) of a "judicial decision" as the result of actions aiming at using a certain subset of legal language ("expressions" resulting from linguistic constructions realized by syntactic, lexical, and textual forms) in order to assign ("considering", "qualifying") legally-relevant entities ("qualified" entities or *tokens*), with appropriate *types* ("judged as" such).

For example, a legal rule may express the situation, by which a legal consequence is "applied" for "judging" a certain legal status "considered" by that legal rule.

Now, using the D&S framework [20], we can conceptualize judicial decisions in the given example as follows: a legal consequence is a description of a situation involving a legal status, whereas the legal rule is a description of the reasons (another situation) why the "typing" is applied in order to judge the considered fact.

This *layering* of descriptions and situations is a typical feature of D&S that allows to overcome the ambiguity of extensional "typing", as in the case of a class "Person" used to type a person, versus intensional "typing" as in the case when we want to take a different perspective on a fact or entity, i.e. what the legal consequence case is about.

From the logical viewpoint, this is non-trivial because we need at the same time to *refer* to a class, as well as to *talk* about that class [21].

Remarkably, the classes we (re)define here are actually reused from the LKIF-Core ontology, in order to align our pattern to it. In fact, LKIF-Core does not attempt to axiomatize the intuition we have just described, and limits itself to distinguishing two "mental objects": `lkif:Legal_Expression` and `lkif:Qualification`, and a role-like class: `lkif:Qualified`. Our alignment is compliant to LKIF-Core, but adds a relational pattern to make sense of judicial decisions, and possibly other legal notions that can be conceptualized in the D&S framework (cf. [22]).

`Qualifying_Legal_Expression` is a kind of `lkif:Legal_Expression`, and includes legal expressions that ascribe a legal status to a person or an object. For example:

- *x is a citizen*
- *x is an intellectual work*
- *x is a technical invention*

The `lkif:Qualification` class includes legal acts (e.g. contractual agreements, judgements) that produce *qualifying legal expressions*. In the examples above, the acts producing the sentences "*x* is a citizen", "*x* is an intellectual work", and "*x* is a technical invention" are *qualifications*.

Legal acts are typically *speech acts* (cf. [3]) influencing the behavior of people and institutions by means of the performative or normative value of the meaning they express (semiotics.owl [7] is an ontology design pattern formalizing speech acts in OWL, see also [19] for an application to legal ontologies).

The modeling of qualifying legal expressions also takes into account Searle's theory of constitutive acts, and the distinction between *fact-tokens* and *fact-types* (see [49]), which is modeled following D&S as situations vs. descriptions. The generalization over the entities that can be qualified is provided by `lkif:Qualified`, a class including anything that is object of a qualification.

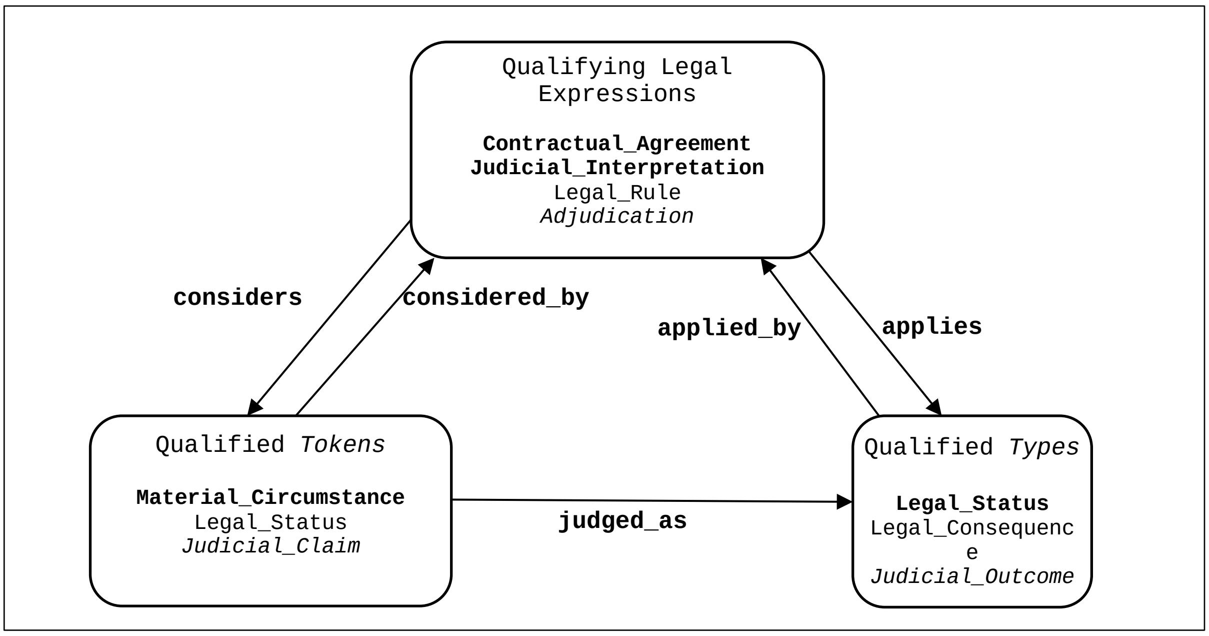


Fig. 4. Qualifications produce expressions about how some legal entity (token qualified entity) is qualified by a type. A type can also be qualified on its turn, as conceptualized in the *qualification pattern*. For each class, some subclasses are provided as examples. The `judged_as` property is defined through a property chain of the main properties `considered_by` and `applies`.

[7] http://www.ontologydesignpatterns.org/cp/owl/semiotics.owl

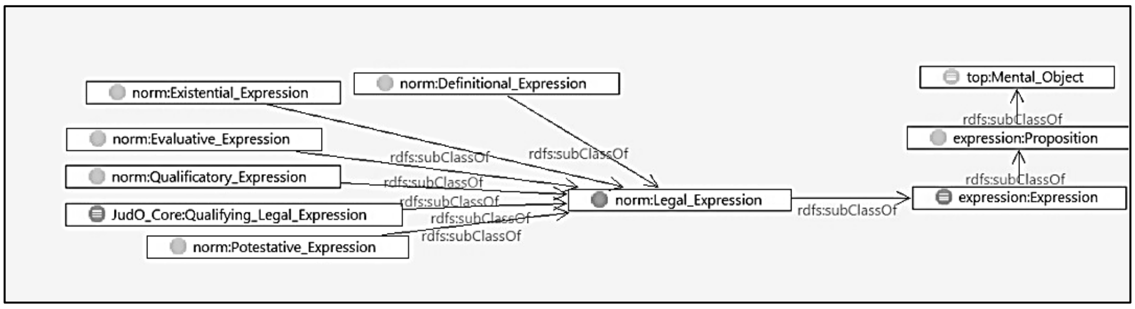

norm:Existential_Expression
norm:Definitional_Expression
top:Mental_Object
norm:Evaluative_Expression
expression:Proposition
norm:Qualificatory_Expression
JudO_Core:Qualifying_Legal_Expression
norm:Legal_Expression
expression:Expression
norm:Potestative_Expression
rdfs:subClassOf

Fig. 5. Taxonomical relations for the Legal_Expression class.

In the examples, both “*x*” (e.g. a *material circumstance* or *legal fact*), and its *types* (e.g. *citizen*, *intellectual work*, *technical invention*) are *qualified* elements, because a qualification tells us something about *x*, but at the same time it provides an example of *citizen*, *intellectual work*, or *technical invention*. In formal ontology, this means that qualifications provide both *instantiation* and *exemplification* [32]. In cognitive science, this means that qualifications introduce both a *categorization*, and a *prototype* [46].

Since the main object to be represented in JudO is the normative/judicial qualification brought forward by performative utterances (contractual agreements, legal rules and – most important – judicial interpretations), the classes presented above constitute the nucleus of the judicial core ontology. As we anticipated, the three classes constitute an ontology design pattern [19] specializing a part of the D&S framework [18,20]: *qualifications* are speech acts that produce descriptions (expressed by *qualified legal expressions*) that characterize qualified elements (either at the instance or type level), and can describe relevant *legal situations* when legal performatives and norms are applied to the social world.

From the design viewpoint, the *qualification design pattern* (Figure 4) defines two further object properties: ***considers*** and ***applies*** (with their inverse properties ***considered_by*** and ***applied_by***). The first one, *considers*, represents the relations between qualifications and instance-level qualified elements (e.g. a judicial interpretation *considers* a material circumstance). The second property (*applies*) represents relations between qualifications and type-level qualified elements (e.g. a judicial interpretation *applies* a legal consequence to categorize and exemplify a material circumstance).

Considering that qualifications are also expressed by qualifying legal expressions, they are designed as a reification of a ternary relation that in first-order logic would be represented e.g. as **qualifies**(*exp*, *obj*, *type*), with **QualifiedLegalExpression**(*exp*), **QualifiedInstance**(*obj*), and **QualifiedType**(*type*). The *Descriptions and Situations* framework provides a vocabulary to the well-known n-ary reification pattern, enabling to model both entities and concepts in the same first-order model. The availability of “punning” in OWL2 helps managing this meta-level flavor (see [21] for a detailed analysis of design alternatives with n-ary relation reification and the *Descriptions and Situations* patterns).

The qualification pattern can be used for different scenarios:

- A `Contractual_Agreement` *considers* a `Material_Circumstance` and *applies* a `Legal_Status;`
- A `Judicial_Interpretation` *considers* a `Material_Circumstance` and *applies* a `Legal_Status`;
- A `Legal_Rule` *considers* a `Legal_Status` and *applies* a `Legal_Consequence`;
- An `Adjudication` *considers* a `Judicial_Claim` and *applies* a `Judicial_Outcome`.

3.2.2. Construction of the Qualifying Expression class in LKIF-Core

LKIF-Core [30] is an established generic legal ontology, and as we have mentioned, JudO is compatible to it. In this section we explain some of the measures taken to obtain this compatibility.

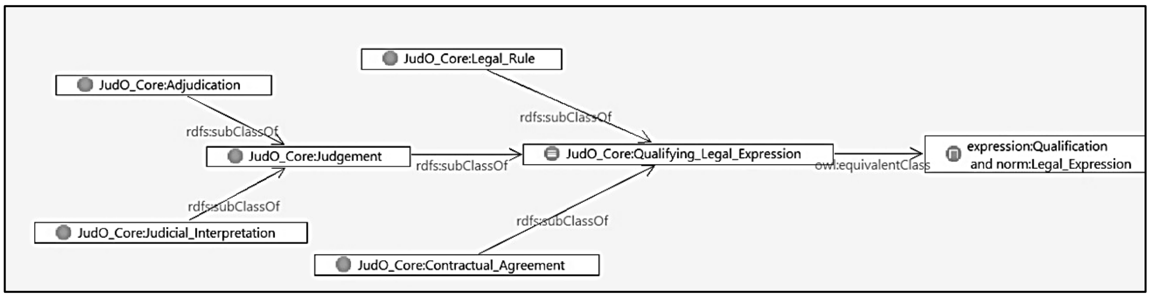


Fig. 6. Taxonomical relations for the `Qualifying Legal Expression` class.

The `Qualifying_Legal_ Expression` class (Figure 5) is aligned to `lkif:Qualificatory_Expression`, but is characterized in JudO as the intersection of `lkif:Legal_Expression` (Figure 6) and `lkif:Qualification` (Figure 7). This choice is due to the desire of being robust against the pervasive ambiguity between expressions and the descriptions they express. It is in fact quite difficult to find evidence of entities that are purely an expression or a description.

JudO also specializes the `lkif:qualifies` property into `considers` (modeled as a superclass of the LKIF-Core properties *evaluates*, *allows*, *disallows*) and `applies`.

The `Qualifying_Legal_Expression` class represents dispositions, which in the sample case are the three legal expressions used in contract law-related judicial decisions: `Contractual_Agreement`, `Legal_Rule` and `Judgement`.

Since `Qualifying_Legal_ Expression` is a subclass of `lkif:Legal_Expression` (Figure 6), its instances contain information related to their original *speech act*: their semantics binds with *externalization*, the *legal power* and *agents* in order to ensure the representation of all aspects that may come into play when facing a legal issue (e.g. legitimacy of the legislative body/court/legal party, characteristics of the corresponding legal document, identity/characteristics of people/bodies involved, etc.). Their main properties are `medium` and `attitude` (see below for a specification of the LKIF-Core `Medium`, `Attitude` and `Agent` classes).

As `Qualifying_Legal_ Expression` is also a subclass of `lkif:Qualification` (Figure 7), its instances contain the information related to the effects they have in the legal world: the legal categories/obligations/effects they create, modify or repeal.

The `lkif:Qualification` and `lkif:Qua-lified` classes are originally linked only by a single property `lkif:qualifies` (inverse `lkif:qualified_by`), but in order to represent this conceptualization, the object property `lkif:qualifies` has been aligned as a super property of two JudO properties: `considers` and `applies`, representing the *object* (instance-level) and the *destination* (type-level) of the qualification respectively.

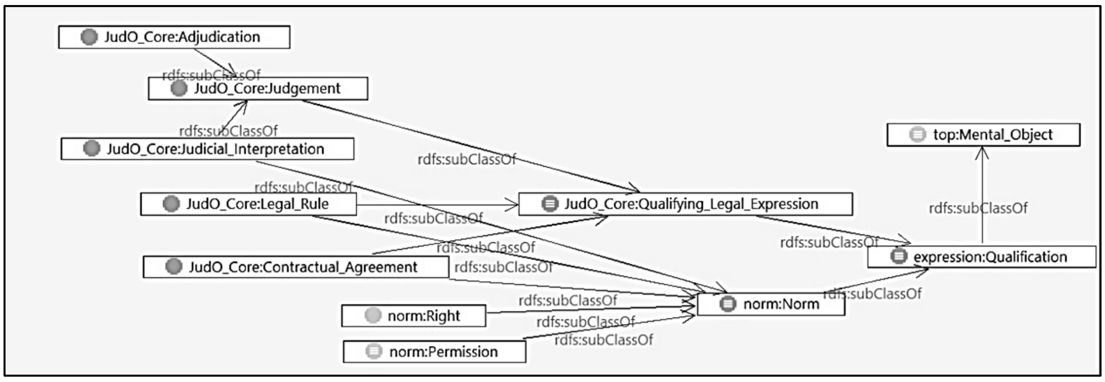


Fig. 7. Taxonomical relations for the `Qualification` class.

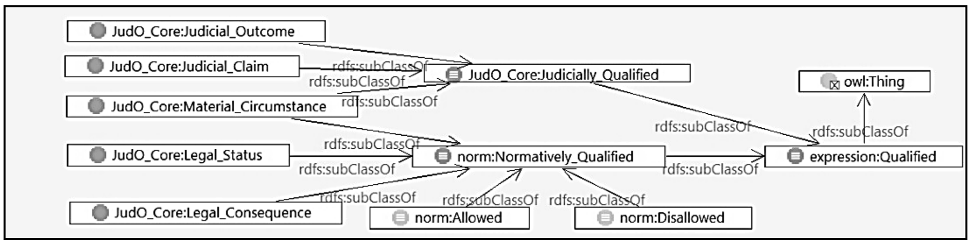

JudO_Core:Judicial_Outcome
JudO_Core:Judicial_Claim
JudO_Core:Material_Circumstance
JudO_Core:Legal_Status
JudO_Core:Legal_Consequence
JudO_Core:Judicially_Qualified
norm:Normatively_Qualified
norm:Allowed
norm:Disallowed
expression:Qualified
owl:Thing
rdfs:subClassOf

Fig. 8. Taxonomical relations for the `Qualified` class.

3.2.3. Qualified Expressions

The `considers` and `applies` properties range on the `lkif:Qualified` class (Figure 8), whose subclasses include now `lkif:Normatively_Qualified`, and `JudO:Judicially_Qualified`.

The `Normatively_Qualified` class include as subclasses `Material_Circumstance`, `Legal_Status` and `Legal_Consequence`. They represent the expressions that can be directly bound to a `Norm`: while `Material_Circumstance` represents any fact or act that is taken into consideration by a `Norm`, `Legal_Status` represents an institutional fact (e.g. fulfillment of contract, oppressive clause, contract breach) that is normally `considered_by` a `Legal_Rule` and `applied_by` a `Contractual_Agreement` or a `Judgement`. Please note that the link between a `Contractual_Agreement`, and the `Legal_Status` it `applies`, is a “weak” link until a `Judicial_Interpretation` has confirmed it. Finally, `Legal_Consequence` represents the sanction provided by the law in the presence of some `Legal_Status` or `Material_Circumstance`. It covers all cases when the `Legal_Rule` `considers` some `Normatively_Qualified` expression, but does not simply `allows`, `disallows` or `evaluates` it.

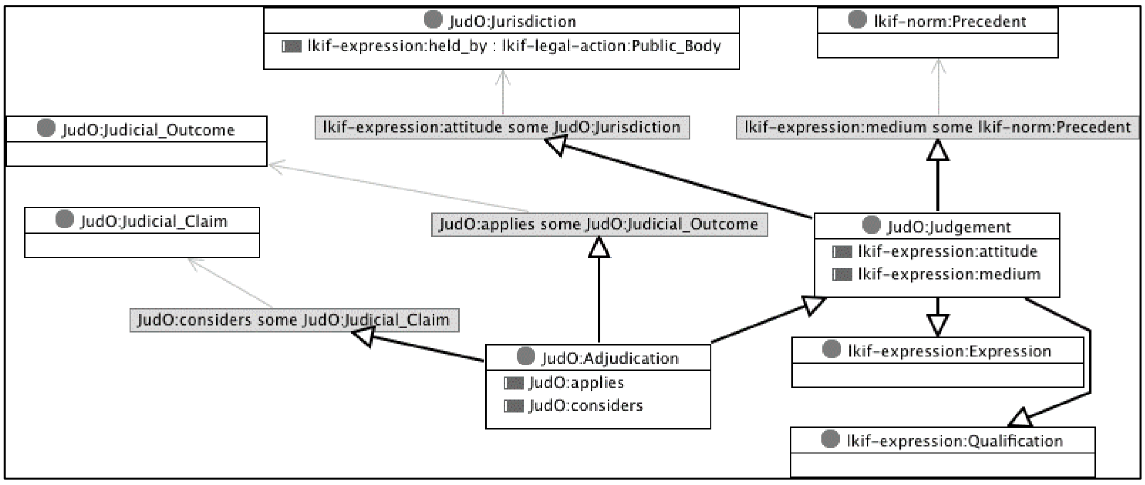

JudO:Jurisdiction
lkif-expression:held_by : lkif-legal-action:Public_Body
lkif-norm:Precedent
JudO:Judicial_Outcome
lkif-expression:attitude some JudO:Jurisdiction
lkif-expression:medium some lkif-norm:Precedent
JudO:Judicial_Claim
JudO:applies some JudO:Judicial_Outcome
JudO:Judgement
lkif-expression:attitude
lkif-expression:medium
JudO:considers some JudO:Judicial_Claim
JudO:Adjudication
JudO:applies
JudO:considers
lkif-expression:Expression
lkif-expression:Qualification

Fig. 9. Visualization of the `adjudication` class and of its semantic connections.

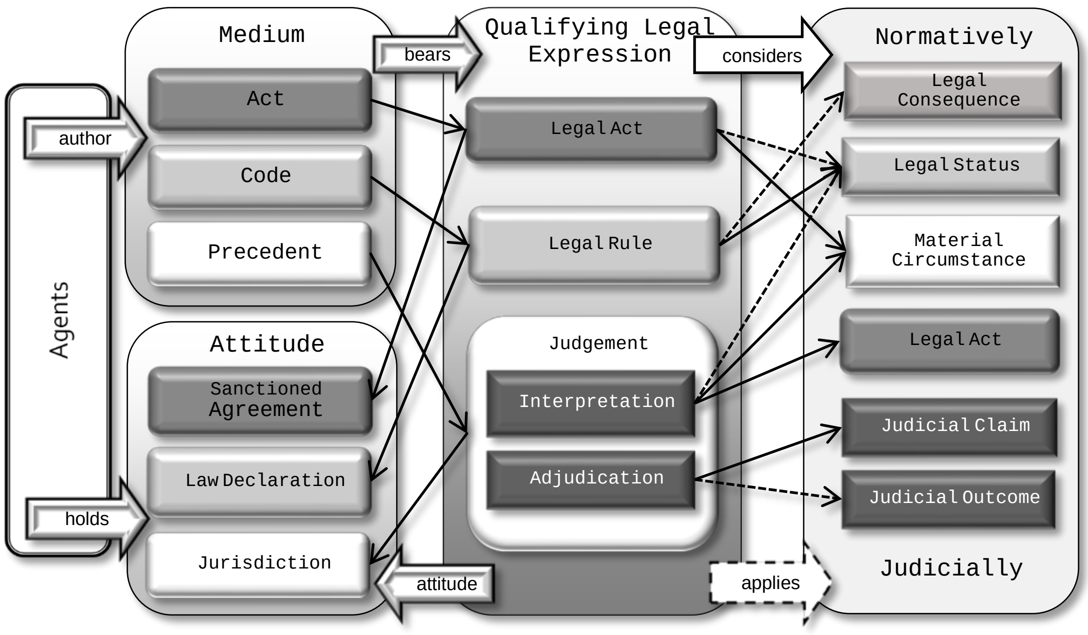


Fig. 10. The Core Ontology graph: boxes represent classes; continuous arrows represent either the *bears*, *attitude* or *considers* properties; dashed arrows represent the *applies* property.

`Judicially_Qualified` expressions include `Judicial_Claim`, `Judicial_Outcome` and all elements taken into consideration during a legal proceeding (e.g. `Contractual_Agreeement`, but also `Legal_Rule`, especially in Cassation Court and Costitutional Court sentences). `Judicial_Claim` is the claim of the legal proceeding. It is `considered_by` an `Adjudication`: the answer of the judge to the claim (subclass of `lkif:Qualification`). The content of an answer (rebuttal/acceptation of the claim or any other possible outcome foreseen by the law) is represented by an instance of the `Judicial_Outcome` class, `applied_by` an `Adjudication`. Therefore, the representation is the following: a `Judicial_Claim` is `considered_by` an `Adjudication` that `applies` a `Judicial_Outcome`.

`Judicially_Qualified` expressions include `Judicial_Claim`, `Judicial_Outcome` and all elements taken into consideration during a legal proceeding (i.e. `Contractual_Agreeement`, but also `Legal_Rule`, especially in Cassation Court and Costitutional Court sentences). A `Judicial_Claim` is the claim of a legal proceeding. It is `considered_by` an `Adjudication`, the answer of the judge to the claim (subclass of `judo:Judgement < lkif:Qualification`).

The content of the answer (rebuttal/acceptation of the claim or any other possible outcome foreseen by the law) is represented by the `Judicial_Outcome` class, `applied_by` the `Adjudication`. The resulting representation is that a `Judicial_Claim` is `considered_by` an `Adjudication` that `applies` a `Judicial_Outcome` (Figure 9).

3.2.4. The `judged_as` *Property Chain*

The aspects taken into consideration during a legal proceeding are included in the `Judicially_ Qualified` class as long as they are actually `considered_by` some `Judicial_Interpreta- tion`. For example, a `Contractual_Agreement` can be `considered_by` a `Judicial_ Interpretation` that `applies` some `Legal_ Status` to it (e.g. the agreement can be *oppressive*, *inefficacious*, can *represent an arbitration clause*, can be *specifically signed by both parties*, …). In such cases, an OWL2 property chain directly links a `Contractual_Agreement` to the `Legal_ Status` judicially applied to it. This property, called `judged_as`, enriches the judicial qualification ontology design pattern presented above.

3.2.5. Media, Propositional Attitudes and Agents

Some LKIF-Core properties and classes support the representation of the context of an `Expression`.

The `Medium` class identifies the support, through which a proposition is expressed. In JudO, the `medium` property has not been used to represent the

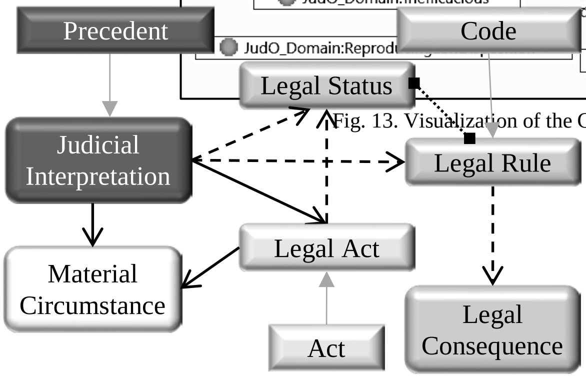


Fig. 11. Semantic relations between represented knowledge. Grey arrows represent the `bears` property, continuous arrows represent the `considers` property, dashed arrows represent the `applies` property. The connection from legal statuses to legal rules is ensured through a qualified class (see 3.3.1.).

material support of an `Expression`, but rather its *genus* (its textual source: `Contract`, `Precedent`, `Code`).

The `lkif:Propositional_Attitude` class has been used as a superclass of `Jurisdiction`, `Law_Declaration` and `Agreement`, in order to represent the enabling powers behind a `Judgement`: a `Legal_Rule` or a `Contractual_Agreement` respectively.

In order to represent the authors of a qualifying legal expression, a generic `lkif:Agent` (or any other agentive class in common ontologies like DOLCE) is sufficient. The knowledge about agents and attitudes is important in some judicial cases, e.g. if a claim is based on the lack of contractual power by one of the parties, or on the identity/characteristics of a party, or on the lack of force by some law or other regulation – which can in turn depend on the lack of legitimacy of one of its authors.

The modeling of roles (already present in LKIF, DOLCE, and other ontologies) is needed to represent critical factors of particular precedents.

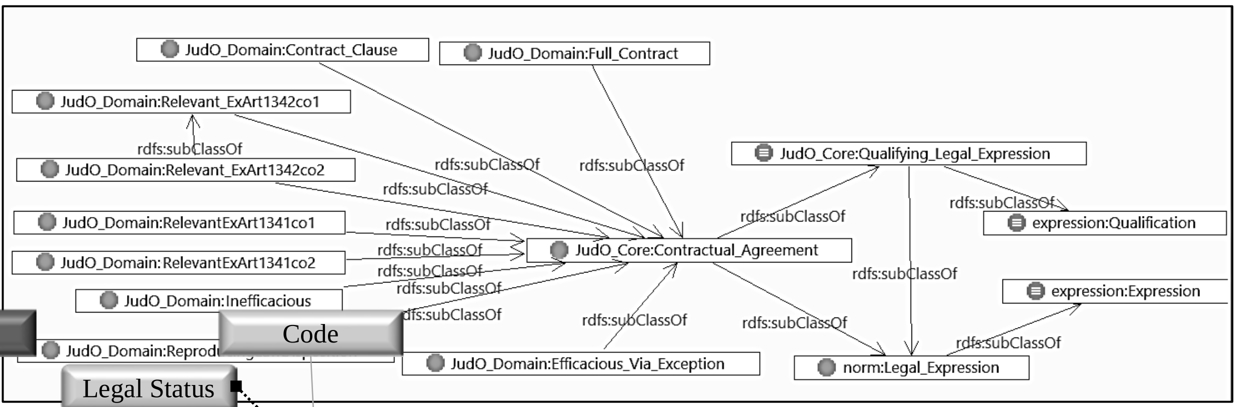

JudO_Domain:Contract_Clause
JudO_Domain:Full_Contract
JudO_Domain:Relevant_ExArt1342co1
JudO_Domain:Relevant_ExArt1342co2
JudO_Domain:RelevantExArt1341co1
JudO_Domain:RelevantExArt1341co2
JudO_Domain:Inefficacious
JudO_Domain:Reprod
JudO_Core:Contractual_Agreement
JudO_Core:Qualifying_Legal_Expression
expression:Qualification
expression:Expression
JudO_Domain:Efficacious_Via_Exception
norm:Legal_Expression
rdfs:subClassOf

Fig. 13. Visualization of the Contractual_Agreement class, including the subclasses introduced by the legal rules.

3.2.6. Modularity of the Core Ontology

JudO is currently oriented to the representation of elements involved in civil-law cases regarding contract law. Nevertheless, JudO provides general – and relatively open – categories for judicial activity

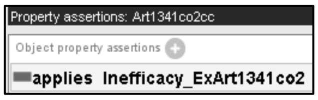

Property assertions: Art1341co2cc
Object property assertions
applies Inefficacy_ExArt1341co2

Fig. 12. Stated property assertion of a Legal Rule instance.

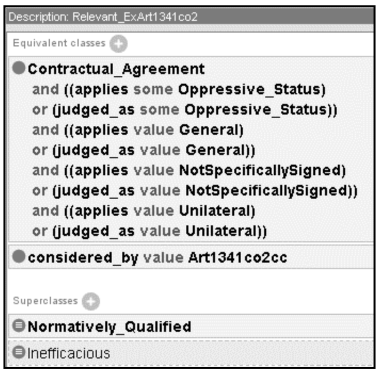
in general, and can be considered as a core to be extended with categorization from other branches of law, since the basic concepts introduced here may come into play also in judgements concerning different subjects.

Figure 10 represents the classes and properties of the core ontology. Figure 11 shows the same information, but provides a simpler intuition of the links between the classes of the ontology.

3.3. Domain Ontology

Following JudO, the metadata taken from judicial documents are represented in the Domain Ontology[6]. The modeling was carried out manually by a legal

[6] https://code.google.com/p/judo/#!



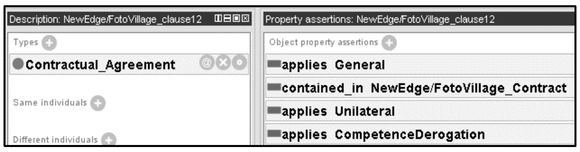
```
Description: Relevant_ExArt1341co2
Equivalent classes
● Contractual_Agreement
   and ((applies some Oppressive_Status)
   or (judged_as some Oppressive_Status))
   and ((applies value General)
   or (judged_as value General))
   and ((applies value NotSpecificallySigned)
   or (judged_as value NotSpecificallySigned))
   and ((applies value Unilateral)
   or (judged_as value Unilateral))
● considered_by value Art1341co2cc
Superclasses
● Normatively_Qualified
● Inefficacious
```

Fig. 14. Axiom for the classification of Contractual Agreements under the legal rule Art. 1341 comma 2.

expert, which actually represents the only viable choice in the legal domain, albeit giving rise to bottleneck issues (see below 5.3.1.). Also, building a legal domain ontology is similar to writing a piece of legal doctrine, thus it should be manually achieved in such a way as to maintain a reference to the author of the model, following an open approach (i.e. allowing different models of the same concept by different authors).

3.3.1. Modelling of laws

The laws involved in the considered domain are represented into the ontology in a quite complex fashion, in order to allow full expressivity of their deontic powers. First of all, they are represented as instances of the `Legal_Rule` class, whose only stated property is to apply the `Legal_Consequence` indicated in the head of the legal rule (Figure 12). A reasoner can infer knowledge about the legal rule, linking it (through the `considers` property) to the material circumstances that fall under the scope of that norm. Material circumstances under the scope of legal rules are also represented, through subclasses of the `Normatively_Qualified` class, according to the template `Relevant_Ex<rulename>` (Figure 13, *ex* is the Latin preposition for indicating a source). An axiom stating the requirements for a material circumstance to be relevant under a legal rule is included in the description of the class, as well as an axiom linking each of its instances to the legal rule through the property `considered_by` (Figure 14). Please notice that in the example (which concerns consumer contracts) these `Relevant_Ex <rulename>` classes are classified under `Contractual_Agreement`, because the effect of the legal rule in this context is to enrich the definition of `Contractual_ Agreement`, adding subdivisions that depend on the legal framework created by the legal rules of the domain.



```
Description: NewEdge/FotoVillage_clause12      Property assertions: NewEdge/FotoVillage_clause12
Types                                          Object property assertions
● Contractual_Agreement                        applies General
                                               contained_in NewEdge/FotoVillage_Contract
Same individuals                               applies Unilateral
Different individuals                          applies CompetenceDerogation
```

Fig. 15. Description and property assertions of the contract clause's content.

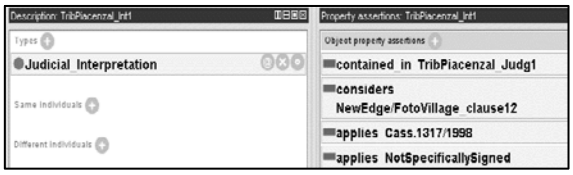
3.3.2. Modeling of contracts

A contract is a composition of one or more `Contractual_Agreements` (a `Contract` for the whole, and multiple `Contract_Clauses` for its parts, an example being provided in Figure 15), each of which represents an obligation arising from the contract. All components of the contract share the same `Attitude` (the “meeting of minds” between the `Agents`) and `Medium` (the kind of support in which the expression is contained). A `Contractual_Agreement` normally `considers` some `Material_Circumstance` and `applies` some `Legal_Status` to it.

In the actual model, the material circumstances considered by the contractual agreement were not included, because the legal effects of the clause have no relevance when capturing the sheer interpretation instances these agreement undergo: it would rather become useful when delving deeper into semantic representation of interpretations.

3.3.3. Modeling of judicial decisions

The instances of the `Judgement` class include an identifier of the case as a whole (the precedent) and several other identifiers for its parts: at least an `Adjudication`, and one or more `Judicial_Interpretations` (Figure 16). They share a common `Attitude` (a `Jurisdiction` power), a `Precedent` medium and some agents



```
Description: TribPiacenzal_Int1                Property assertions: TribPiacenzal_Int1
Types                                          Object property assertions
● Judicial_Interpretation                      contained_in TribPiacenzal_Judg1
                                               considers
Same individuals                                 NewEdge/FotoVillage_clause12
                                               applies Cass.1317/1998
Different individuals                          applies NotSpecificallySigned
```

Fig. 16. Description and property assertions of the judicial interpretation.

(claimant, defendant, and court). An `Adjudication` contains the `Judicial_ Outcome` of the `Judicial_Claim` (it `considers` the claim and `applies` the outcome), while a `Judicial_Interpretation` `considers` a `Material_Circumstance` and `applies` one or more `Legal_Statuses` (and zero or more `Precedents`) to it. The precedents cited by a judge in a decision are added directly to an interpretation instance: the reasoner would be able to distinguish between legal statuses and precedents. Rules expressed by precedents can be modeled similarly as legal rules.

3.3.4. Reasoning on the knowledge base

The consistency of the Knowledge Base was checked with the Hermit 1.3.6 [8] reasoner. In particular, we have used it in order to check if the ontology gives a unique and correct answer to formalized questions (e.g. asking about the validity of some proof, or about the qualification of factual events under legal principles). When a `Contractual_Agreement` (the expression brought by a `Contract_Clause`) is `considered_by` some `Judicial_ Interpretation`, the ontology gathers all relevant information about the documents involved: contract parties, judicial actors, legal statuses applied to the agreement (eventually in comparison to the one suggested by the contract/judicial parties), rules of law relevant to the legal status, final adjudication of the claim, part played in it by the interpreted agreement, etc.

An immediate application of this semantically enriched knowledge base consists in performing advanced querying on precedents, but more can be achieved by combining different `Judicial_Interpretations` with knowledge coming from a contract and the related applicable law. The ontology reasoner is in fact capable of predicting – to the extent allowed by the details included in the ontology – the outcome of the judge (that a clause will be judged as valid/invalid), and to run inferences about the agreement (as interpreted, the clause in the example of Figure 17 is relevant for the legal rule contained in article 1342 comma 2 of Italian Civil Code, and inefficacious in the light of the same norm).

[8] http://hermit-reasoner.com/

This inferred knowledge is important for two reasons:

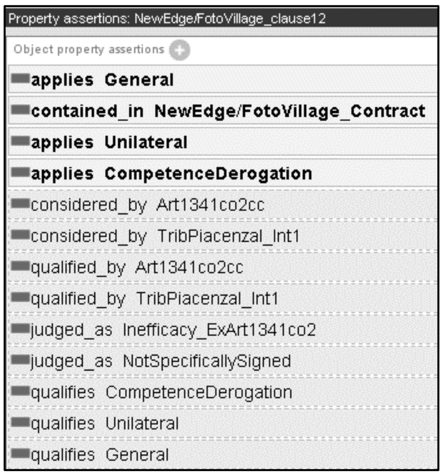
a. by predicting the judge's final statement on the clause (even if not the one on the claim), this knowledge represents a deontic check on the legal consequences the judge takes from its interpretation;

b. it gives a fundamental element for an argumentation system to support the explanation of the adjudication of the claim. The argumentation system, in fact, will be able to use the (stated and inferred) elements of the decision's groundings to support and explain the `Adjudication` contained in the last part of the judgement.

3.4. Some observations on OWL2 patterns

OWL2 [53] is a milestone standard for the Semantic Web, introducing new useful modeling patterns, and is relevant to any project willing to contribute to the "Web of Data"). In the case of legal reasoning, it is important to assess to what extent we can use those new patterns.

Also, we want to assess how OWL2 could help designing the background for the application of defeasible logic: OWL is not designed for managing defeasibility directly, being only able to capture the monotonic knowledge to be reused in the rule layer; nevertheless, the gap between ontology and rules is often underestimated, and the benefits coming from OWL2 have not yet been considered in detail. The



```
Property assertions: NewEdge/FotoVillage_clause12
Object property assertions
applies  General
contained_in  NewEdge/FotoVillage_Contract
applies  Unilateral
applies  CompetenceDerogation
considered_by  Art1341co2cc
considered_by  TribPiacenzal_Int1
qualified_by  Art1341co2cc
qualified_by  TribPiacenzal_Int1
judged_as  Inefficacy_ExArt1341co2
judged_as  NotSpecificallySigned
qualifies  CompetenceDerogation
qualifies  Unilateral
qualifies  General
```

Fig. 17. Inferred knowledge on the Contractual Agreement instance.

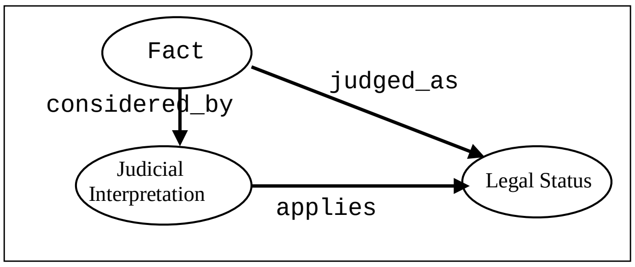


Fig. 18. The property chain judged_as.

assumption here is that the more we can use classical inferences, the better.

OWL2 introduces several features enriching the original Ontology Web Language, enabling a richer or simpler representation of knowledge, especially when dealing with meta-level assertions, properties and datatypes.

Some of the new features have been tested in JudO and the contract ontology, with varying results from the design perspective. For example, disjointness between properties can be used to reason about legal statuses, but it is necessary to create as many properties as possible statuses, which would negatively affect the readability of the ontology.

Other new constructs resulted to enhance expressivity without negatively affecting design and usability of the ontologies. We describe them by using OWL functional syntax.

3.4.1. Property chains

The OWL2 construct `ObjectPropertyChain`, used within a `SubObjectPropertyOf` axiom, allows a property to be inferred from the composition of several properties. JudO relies on one particular property chain useful in the judicial domain. The property chain:

```
considered_by o applies
SubObjectPropertyOf judged_as
```

is exemplified in Figure 18 with respect to judicial interpretations, but is defined on all cases of chains thorugh the `considered_by` and `applies` properties, e.g. in rule applications in order to create a direct interpretational link between a *material circumstance* and a *legal status*. In practice, in the first case, a `Judicial_Interpretation` `considers` a `Material_Circumstance` and `applies` a `Legal_Status`, the `judged_as` property chain in inferred, and creates a direct link between the circumstance and its status.

In the second case, a legal rule axiom works through an “anonymous qualified class” (see 3.3.1.), which links all relevant expressions to the legal rule instance through the `considered_by` property, and the legal rule `applies` a legal consequence. The `judged_as` property chain unifies the two properties (from a qualified expression to a law, and from a law to a legal consequence), and brings their semantics to the surface by creating a direct property linking the contract clause to its status (`judged_as Inefficacy`). OWL2 property chains could actually lead to an even more direct and complete solution, by removing the need for the anonymous subclass in order to identify the clause instances `considered_by` the relevant law. In the current version of the ontology, `judged_as` connects a material instance (i.e. a contract clause) to a legal status or legal consequence (i.e. `oppressive`, `inefficacious`) through a judicial interpretation. With the open world approach of OWL, this creates a sprawling of `judged_as` chains being applied to the metadata. Such inferences are correct; nevertheless, they substantially increase the number of triples in the ontology. When reasoning on large data, it is therefore necessary to prune chain-based inferences in order to retain only those that are interesting for the task at hand. Since pruning would eliminate semantic content actually existing in legal documents, it has to be performed depending on the task of the rules application.

3.4.2. Negative object property assertions

A negative object property assertion such as:

```
NegativeObjectPropertyAssertion(OP a b)
```

states that the individual `a` is not linked to `b` by the object property `OP`. E.g. given an ontology including the following axiom:

```
NegativeObjectPropertyAssertion(hasSon
Bob Meg)
```

the ontology becomes inconsistent if it is extended with the following assertion:

```
ObjectPropertyAssertion(hasSon Bob Meg)
```

Negative object property assertions are useful to avoid complicated workarounds for negating assertions. For example, the legal status `NotSpecificallySigned` is needed in OWL in order to state that a certain status is not `SpecificallySigned`, e.g.:

```
EquivalentClasses
 (SpecificallySigned ObjectOneOf
  (NotSpecificallySigned
SpecificallySigned))

DifferentIndividuals
 (SpecificallySigned
NotSpecificallySigned)

ObjectPropertyAssertion
 (applies ContractA
NotSpecificallySigned)
```

while in OWL2 the following construct is sufficient:

```
NegativeObjectPropertyAssertion
 (applies ContractA
SpecificallySigned)
```

*3.4.3. Key*s

A `HasKey` axiom states that each *named* instance of a class is uniquely identified by the set of data or object properties assertions specified in the axiom - that is, if two named instances of the class coincide on values for each of key properties, then those two individuals are the same. This feature is useful to control that the elements in a judicial claim, e.g. the parties, the contract, the norm, and the decision itself, are actually unique for that claim.

3.4.4. Annotation properties

Originally OWL allowed extra-logical annotations to be added to ontology entities, but did not allow annotation of axioms. OWL2 allows annotations on ontologies, entities, anonymous individuals, axioms, and annotations themselves.

This feature is used in the judicial ontology library to provide a full-fledged information structure about the author of each piece of the model (i.e., who modeled a certain axiom, which legal text it refers to, and who/when/how was the original legal text created). Moreover, it is possible to give domains (AnnotationPropertyDomain) and ranges (AnnotationPropertyRange) to annotation properties, as well as organize them in hierarchies (SubAnnotationPropertyOf). These special axioms have no formal meaning in OWL2 direct semantics, but carry the standard RDF semantics in RDF-based semantics, via the mapping to RDF vocabulary, therefore annotation axioms can still be used in queries and in shallow reasoning when needed.

3.4.5. N-ary datatypes

In OWL it was not possible to represent relationships between values for one object, e.g., to represent that a square is a rectangle whose length equals its width. N-ary datatype support was not added to OWL2 because it was “unclear” what support should be added. However, OWL2 includes all syntactic constructs needed for implementing n-ary datatypes. The Data Range Extension: Linear Equations W3C Note[9] proposes an extension to OWL2 for defining data ranges in terms of linear (in)equations with rational coefficients. This kind of equations is important in the process of identifying individuals to classify under a legal ontology framework, based on the comparison between quantitative values corresponding to several factors for a same individual, even if the path-free limitation imposed by the W3C Note (i.e. we cannot compare the same kind of value for two different individuals) is actually a problem for some legal cases, as when comparing the age of a person, and the legal age limit stated by a norm.[10]

3.4.6. Qualified cardinality restrictions

While originally OWL allows for restrictions on the number of instances of a property (i.e. for defining the class of persons that have at least three

[9] http://www.w3.org/TR/owl2-dr-linear/

[10] This modeling issue can be solved by a SPARQL query on RDF data, which is a suboptimal solution with respect to complete reasoning.

children), it does not provide means to constrain object or data cardinality (qualified cardinality restrictions), i.e. for specifying the class of persons that have at least three children, who are girls). In OWL2 both qualified and unqualified cardinality restrictions are possible through the constructs: `ObjectMinCardinality`, `ObjectMaxCardinality`,`ObjectExactCardinality` (respectively `DataMinCardinality`, `DataMaxCardinality`,`DataExactCardinality`). These restrictions are important to enrich the ontology with elements ensuring automatic classifications of qualified properties (e.g. the minimum income needed for a claim to be classified under a certain category).

4. An Example of judgement modeling

JudO domain application is explained here through a simple example of data insertion and knowledge management. The following is a description of the case to be modeled:

In the decision given by the 1st section of the Court of Piacenza on July 9th, 2009,[11] *concerning contractual obligations between two small enterprises ("New Edge sas" and "Fotovillage srl", from now on α and β), the judge had to decide whether clause 12 of α/β contract, concerning the competent judge (Milan instead of Piacenza) could be applied. The judge cites art. 1341 comma 2 of Italian Civil Code that says: "a general and unilateral clause concerning competence derogation is invalid unless specifically signed". In the contract signed by the parties there is a distinct box for a "specific signing" where all the clauses of the contract are recalled by their number. The judge, with the support of precedents (he cites 9 Cassation Court sentences) interprets the "specific signing" as not being fulfilled through a generic recall of all the clauses, and therefore declares clause 12 of α/β contract invalid and inefficacious. The claim of inefficacy of clause 12, brought forward by α, is thus accepted, undercutting the claim of a lack of competence by the judge of Piacenza, brought forward by β, which is rejected.*

[11] Sent. N. 507 del 9 Luglio 2009, Tribunale di Piacenza, giudice dott. Morlini.

In order to represent the knowledge contained in the judgement text, three documents have to be modelled: Art. 1341 comma 2 of Italian Civil Code, the contract between the two enterprises α and β, and the decision by the Court of Piacenza.

4.1. Modelling of the law

The following is the law disposition involved in the judicial decision:

Article 1341 subsection (*comma*) 2 of Italian Civil Code: *Clauses concerning arbitration, competence derogation, unilateral contract withdrawal, and limitations to: exceptions, liability, responsibility, and towards third parties, are inefficacious unless they are specifically signed by writing.*

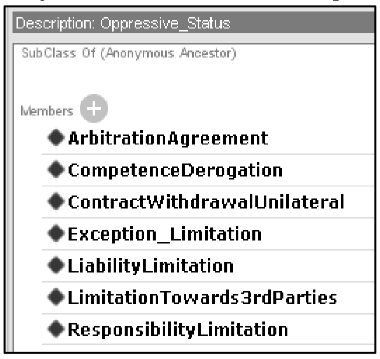
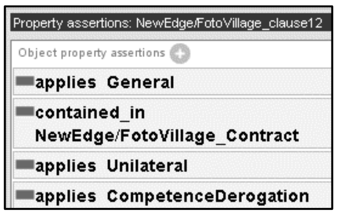

Description: Oppressive_Status
SubClass Of (Anonymous Ancestor)
Members
- ArbitrationAgreement
- CompetenceDerogation
- ContractWithdrawalUnilateral
- Exception_Limitation
- LiabilityLimitation
- LimitationTowards3rdParties
- ResponsibilityLimitation

Fig. 19. The list of legal statuses classified as oppressive.

The disposition is represented as a `Qualifying Legal Expression` (`Legal_Rule`) called "art1341 Co2" (with a `Code` medium, a `Law_Declaration` attitude, and a `Parliament` agent), and the `Qualified` subclass `Relevant_ExArt1341co2`. As seen in 3.3.1, a `Legal_Rule considers` a (combination of) `Legal_Status`(es) and `applies` a `Legal_Consequence`, possibly with a deontic operator. Therefore, any individual that has the characteristics required by the law is `considered_by` the `Legal_Rule`, which in turn `allows`/`disallows`/`evaluates` or `applies` some `Legal_Consequence` to it. In the example of figure 15, each `Contractual_Agreement` which *applies* "General", "Unilateral", "NotSpecificallySigned" and an `Oppressive_Status` (Figure 19) will be `considered_by` "art1341Co2", which in turn `applies` the `Legal_Consequence` of


Property assertions: NewEdge/FotoVillage_clause12

Object property assertions

applies General

contained_in NewEdge/FotoVillage_Contract

applies Unilateral

applies CompetenceDerogation

Fig. 20. Stated property assertions for the sample agreement.

“invalidityExArt1341co2”. The individuals “competentJudge” and “notSpecificallySigned” are thus created as `Legal_Statuses` that can be `considered_by` a `Legal_Rule` and `applied_by` a `Contractual_ Agreement`, and the individual “invalidityExArt1341co2” is created as a `Legal_Consequence` `applied_by` the `Legal_Rule` “art1341Co2”.

4.2. Modelling of the contract clause

The `Contract_Clause` “α/βClause12” (Figure 20) is created and linked to a `Contractual_ Agreement` which `applies` the `Legal_Statuses` of “General”, “Unilateral” and “Competence Derogation”. This is done because there is no argue between the parties about whether clause 12 concerns a competence derogation. However, as explained before, this kind of link is a *weak* one, considering that contractual parties have no power to force a legal status into a contract, and that assigning a contractual agreement to the legal figure it evokes is the main activity brought forward by judicial interpretation in the contracts field. For this reason, the property `applies` related to a `Legal_Status` is weak when its domain is a `Contractual_Agreement`, and prone to be overridden by a contrasting application performed by a `Judicial_Interpretation`.

4.3. Modelling of the judicial interpretation

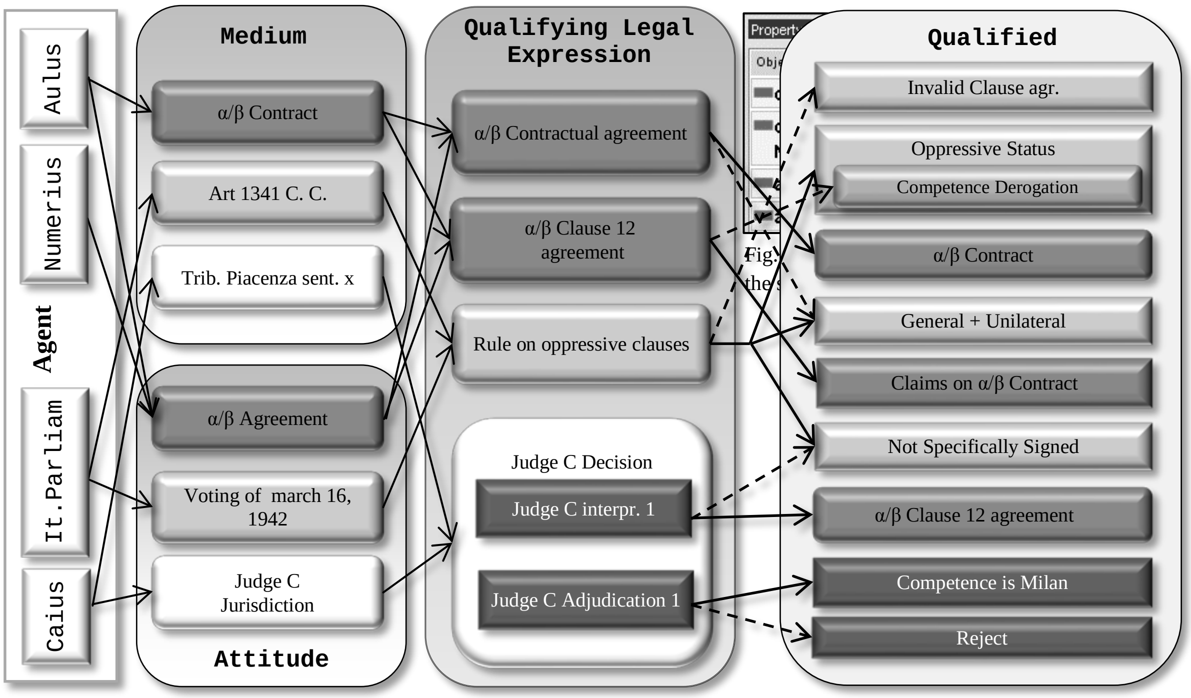
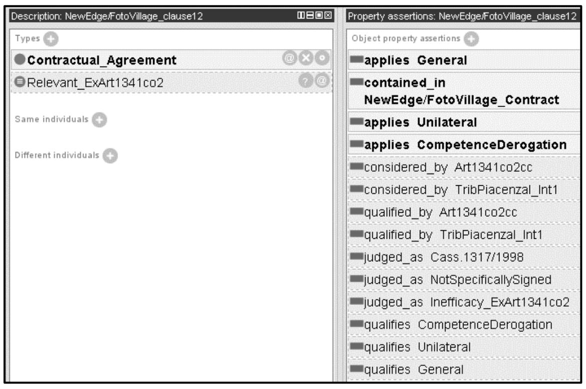


Fig. 22 - The graph showing the model of the sample case. The general classes of Figure 10 have been substituted with the sample instances. The properties (arrows) connect the same classes of the core ontology.



```
Description: NewEdge/FotoVillage_clause12
Types
Contractual_Agreement
Relevant_ExArt1341co2
Same individuals
Different individuals

Property assertions: NewEdge/FotoVillage_clause12
Object property assertions
applies General
contained_in NewEdge/FotoVillage_Contract
applies Unilateral
applies CompetenceDerogation
considered_by Art1341co2cc
considered_by TribPiacenzal_Int1
qualified_by Art1341co2cc
qualified_by TribPiacenzal_Int1
judged_as Cass.1317/1998
judged_as NotSpecificallySigned
judged_as Inefficacy_ExArt1341co2
qualifies CompetenceDerogation
qualifies Unilateral
qualifies General
```

Fig. 23. Inferred Description and property assertions for the contract clause content.

The `Judgement` instance is created, as well as its components (individual interpretation instances, adjudication, etc.). Among them, the "TribPiacenzaI_Int1"`Judicial_Interpretation` is created (Figure 21): it `considers` the `Contractual_Agreement` contained in "α/βClause12", and `applies` the "notSpecifically Signed" `Legal_Status`. The instance contains also a reference to the precedent (Cass.1317/1998), which represents a semantically-searchable information on the interpretation instance. Figure 22 summarizes all the elements created for the various classes, and the relations among them.

4.4. *Reasoning on the knowledge base*

In the example, when all the relevant knowledge is represented into the ontology, an OWL2 reasoner is capable of inferring that "The agreement contained in clause 12 of the α/β contract is invalid ex article 1341 comma 2" (Figure 23). As already explained, this result is reached through a subclass of the `Contractual_Agreement` and `Qualified` classes, defined by an axiom representing the rule of law. Clauses that fulfil the axiom are automatically classified in that class, and thus `considered_by` the appropriate law. At this point, the `judged_as` property chain links the clause to the legal consequence, through the legal rule (the clause is `considered_by` the law that `applies` a legal consequence, then the clause is `judged_as` the legal rule). The `judged_as` property gives the clause its final (efficacy/inefficacy) status under that law. Figure 24 explains the whole process as a list of axioms verified by the ontology reasoner.

5. Evaluation of the JudO ontology library

The ontology library, tested on a sample taken from real judicial decisions, meets the following requirements:

- **Text-to-knowledge morphism**: the ontology can correctly classify all instances representing fragments of text. The connection to the Akoma Ntoso markup language ensures the identification and management of those fragments, and of the legal concepts they contain.
- **Distinction between document layers**: The `Qualifying_Legal_Expression` class is the hub entity in the core ontology pattern, being the reification of a n-ary relation at the core of the required reasoning. Its instances can refer to the same text fragment, yet they represent different (and potentially inconsistent) interpretations of that text, as expected in the legal domain. Moreover, the `lkif:Medium` class allows to represent possible different manifestations of the same expression;
- **Shallow reasoning on judgement's semantics**: the Domain Ontology can perform reasoning on the relevance of a material circumstance under a certain law. The property chain `judged_as` and the axioms for law relevance and legal consequence application allow the reasoner to complete the framework,

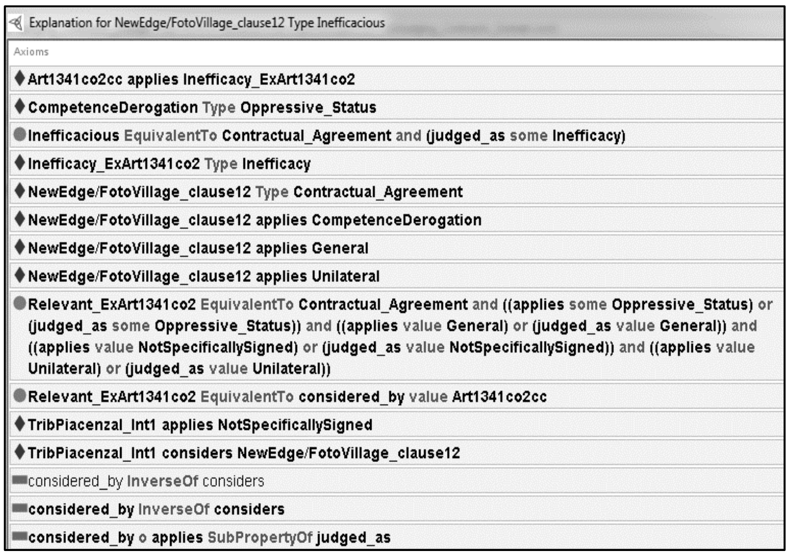

```
Explanation for NewEdge/FotoVillage_clause12 Type Inefficacious
Axioms
♦Art1341co2cc applies Inefficacy_ExArt1341co2
♦CompetenceDerogation Type Oppressive_Status
●Inefficacious EquivalentTo Contractual_Agreement and (judged_as some Inefficacy)
♦Inefficacy_ExArt1341co2 Type Inefficacy
♦NewEdge/FotoVillage_clause12 Type Contractual_Agreement
♦NewEdge/FotoVillage_clause12 applies CompetenceDerogation
♦NewEdge/FotoVillage_clause12 applies General
♦NewEdge/FotoVillage_clause12 applies Unilateral
●Relevant_ExArt1341co2 EquivalentTo Contractual_Agreement and ((applies some Oppressive_Status) or
 (judged_as some Oppressive_Status)) and ((applies value General) or (judged_as value General)) and
 ((applies value NotSpecificallySigned) or (judged_as value NotSpecificallySigned)) and ((applies value
 Unilateral) or (judged_as value Unilateral))
●Relevant_ExArt1341co2 EquivalentTo considered_by value Art1341co2cc
♦TribPiacenzaI_Int1 applies NotSpecificallySigned
♦TribPiacenzaI_Int1 considers NewEdge/FotoVillage_clause12
▬considered_by InverseOf considers
▬considered_by InverseOf considers
▬considered_by o applies SubPropertyOf judged_as
```

Fig. 24. Explanation for the sample agreement being inefficacious.

facilitating the modeling of the knowledge contained in judicial decisions, and thus supporting tools for semi-automatic legal knowledge extraction;

- **Querying**: the `considers/applies` properties allow complex querying on the knowledge base, and the `judged_as` shortcut provides a simplified path in this perspective. Complete formal querying based on temporal parameters is not yet possible, but solutions are envisaged through emerging standards for rules such as LegalRuleML. A basic RDF querying via SPARQL can be used though.
- **Modularity***:* the layered (core/domain) structure of the ontology library renders domain ontologies independent between each other - and yet aligned and integrated, through their compliance to the core ontology.
- **Supporting text summarization**: the ontology library supports the identification of dispositions and decision’s groundings inside a judicial decision.
- **Supporting case-based reasoning**: An argumentation system has been built on top of a lighter version of the ontology library. The axioms concerning law relevancy and law application were removed from the ontology and moved to the rules layer, in order to have them applied not only to the ontology library's knowledge base, but also on the new knowledge derived from the application of the rules. Results of this can be found in [14].

Computability was not an issue in the last ontology library version (<5 seconds reasoning time on a Intel i5@3.30 Ghz), while the Carneades reasoner was moderately encumbered by the application of the rules to the ontology (8-15 seconds in the example described in [15]). This could be improved by optimizing the reasoner and/or with a further refinement of the ontology/rules structure.

5.1. Comparison to related work

The framework presented in this paper relies on previous efforts of the community in the field of legal knowledge representation [10] and rule interchange for applications in the legal domain [27]. The issue of implementing logics to represent judicial interpretation has already been faced in [9,22], albeit only for the purposes of a sample case, and in [54] on a realistic regulation knowledge base, but using a lighter description logic.

The methods applied for the construction of the core legal ontology are similar to those used for [12], an online repository of legal knowledge to provide answers to issues related to legal procedures. The main difference between the two approaches is that the latter relies on application of NLP techniques to user-generated questions in order to return the correct answer. The judicial ontology instead extracts information from official legal documents (laws, decisions, legal doctrine), whose content classification requires the intervention of a legal expert. Furthermore, the ontology in [12] focuses on legal procedure, while the present ontology concerns mainly judicial interpretations carried out by the judge in a decision, seen as subsumptions of material facts or circumstances under abstract legal categories.

The project presented in [48] focuses on a lower layer of the Semantic Web, concerning document structure and data interchange between different legal documents. For the same purposes, the present project relies on Akoma Ntoso (see 3.1.). Besides its being foucused on administrative procedures, the project in [48] shows a rather interesting view on the procedural aspects of legal phenomena, which is something this ontology does not achieve, being this task demanded to an argumentation layer placed on top of the ontology layer.

[17] presents a semi-automatic construction of an ontology concerning the language of a legislative text. The project is focused on the linguistic aspects, in particular on the use of NLP techniques to normalize and formalize the text in a set of concepts previously modeled in an ontology. The ontology is built around DOLCE-based Core Legal Ontology [22] and LRI-Core, which makes it likely to be aligned with the ontology presented in this paper. The ontology in [17], in fact, ensures a close relation with the legal text, even though it does not includes axioms that enable shallow reasoning on specific legal phenomena.

The ontology in [50] is very interesting for the orientation towards NLP, the solid basis on metaphysics, and in that it allows shallow reasoning on a set of simple legal sentences. It is built around the NM ontology ([50], contains a comparison to LRI-Core), and relies on agents to bridge the legal text with the syntax. The approach is interesting, yet the focus on agents somehow overcomplicates the reasoning on complex legal concepts such as that of judicial interpretation. Detecting advanced concepts in legal documents requires in fact a highly complex semantic structure, which prevents reasoning on a large document corpus contents (for a general account on how to model complex legal concepts for automatic detection see [40]). Moreover, as already noted, modelling the dynamics of legal procedure requires a proper implementation of argumentation theory.

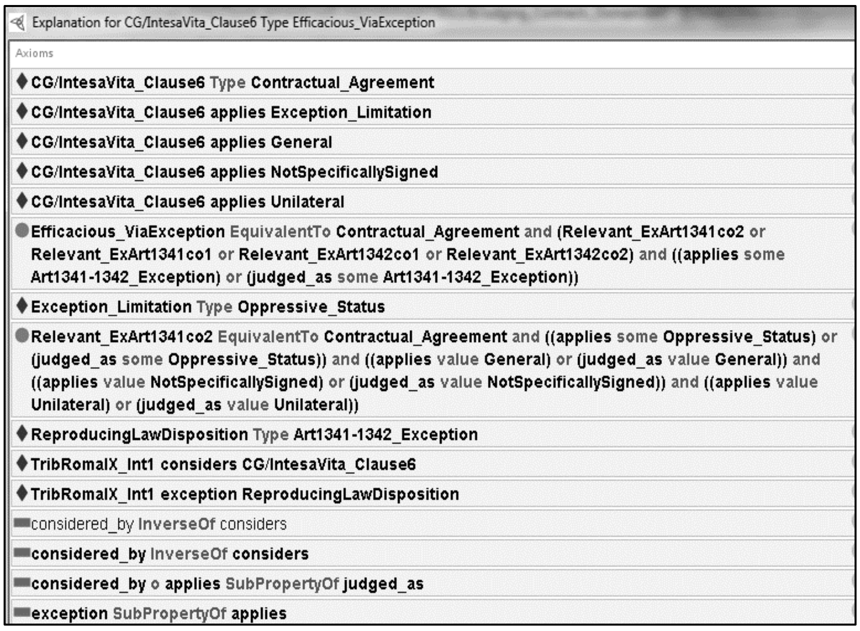
5.2. A bridge towards judicial argumentation

The argumentation system described in [14,15] combines the features of DL-based ontologies with non-monotonic logics such as Defeasible Logics. In particular, T. F. Gordon’s Carneades [25] is based on Walton’s theory [26], and takes into account most of Prakken’s considerations on the subject [44], including argumentation schemes and burden of proof. The Carneades application succeeded in performing the tasks of finding relevant precedents, validating the adjudications and suggesting legal rules, precedents, circumstances that could bring to a different adjudication of the claim.

Many projects tried to represent case-law during the nineties, most of which are related to the work of Kevin Ashley [2]. Their main focus is similar to that assumed in the present research: capturing the elements that contribute to the decision of a judge. They were meant to support legal argumentation teaching in law classes, and the approach was therefore based on concepts rather than on legal documents. No account for the metadata of the original text was given, and there is no ontology underlying the argumentation trees that reconstruct the judge’s reasoning.

Rather than representing a single judicial decision, the approach presented in this paper allows instead to connect knowledge coming from different decisions, and to highlight similarities and differences between them, not only on the basis of factors, dimensions or values, but also on the basis of the efficacy of the legal documents involved (e.g. under temporal and hierarchical criteria). Of course, *templatizing* legal documents is a very complex task, but the intention is not to provide a complete extraction tool, but to create an interface, through which a legal expert can easily identify the legal concepts evoked by the text, and combinations of them, in legal documents.

Deontic defeasible logic systems, such as those presented in [28,31,36], constitute indeed a powerful tool for reasoning on legal concepts. Most of them are explicitly built to import RDF triples, which means that they can perform reasoning on knowledge bases derived from ontologies such as the one presented in this paper. These projects can be placed at an upper layer than the one discussed here: an ontology, in the perspective of the research presented here, focuses on document semantics and basic relations, performing open world reasoning mostly oriented to data completion. Over such


Explanation for CG/IntesaVita_Clause6 Type Efficacious_ViaException

Axioms

♦ CG/IntesaVita_Clause6 Type Contractual_Agreement
♦ CG/IntesaVita_Clause6 applies Exception_Limitation
♦ CG/IntesaVita_Clause6 applies General
♦ CG/IntesaVita_Clause6 applies NotSpecificallySigned
♦ CG/IntesaVita_Clause6 applies Unilateral
● Efficacious_ViaException EquivalentTo Contractual_Agreement and (Relevant_ExArt1341co2 or Relevant_ExArt1341co1 or Relevant_ExArt1342co1 or Relevant_ExArt1342co2) and ((applies some Art1341-1342_Exception) or (judged_as some Art1341-1342_Exception))
♦ Exception_Limitation Type Oppressive_Status
● Relevant_ExArt1341co2 EquivalentTo Contractual_Agreement and ((applies some Oppressive_Status) or (judged_as some Oppressive_Status)) and ((applies value General) or (judged_as value General)) and ((applies value NotSpecificallySigned) or (judged_as value NotSpecificallySigned)) and ((applies value Unilateral) or (judged_as value Unilateral))
♦ ReproducingLawDisposition Type Art1341-1342_Exception
♦ TribRomalX_Int1 considers CG/IntesaVita_Clause6
♦ TribRomalX_Int1 exception ReproducingLawDisposition
▬ considered_by InverseOf considers
▬ considered_by InverseOf considers
▬ considered_by o applies SubPropertyOf judged_as
▬ exception SubPropertyOf applies

Fig. 25. Explanation of a sample contract clause being not inefficacious because of an exception.

knowledge base, rule systems based on non.classical logic dialects can perform more reasoning (as e.g. supported by SPINdle [33]), by importing only the set of triples that best suits their needs. This may be preferable to approaches that try to extend DL to perform defeasible reasoning such as [1]: JudO shows that it is possible to perform classical reasoning while staying within OWL2, while deontic logic, defeasible reasoning and reasoning on argumentation schemes [54] come on top of it.

The same considerations come from the approach in [34], which provides a simple and intuitive way to encode default knowledge on top of terminological KBs.

From this perspective, the idea of deriving a closed-world subset of an OWL2 ontology as presented in [45] seems a good direction, and will in fact be explored, keeping in mind, though, that introducing negation-as-failure in OWL2 is not sufficient to grant the ontology layer the expressivity required for performing argumentation tasks.

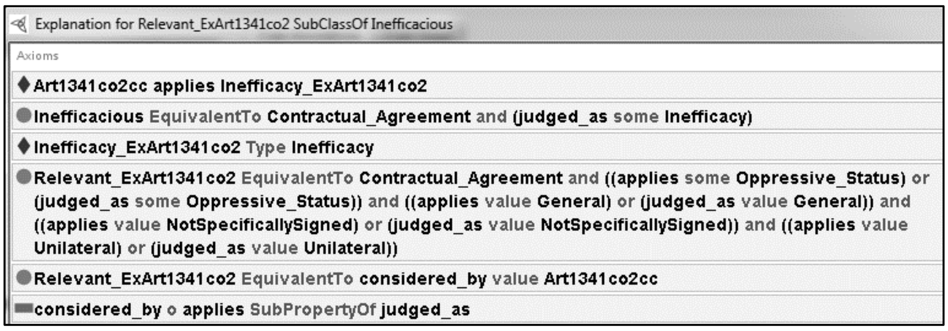
5.3. Issues

5.3.1. The knowledge acquisition bottleneck

Modelling the sample ontology library and extracting knowledge from the sample of case law was carried out manually by a graduated jurist. Similarly, qualified fragments of text under the Akoma Ntoso standard are supposed to be annotated by legal experts. Currently, manual data insertion seems the only viable choice in the legal domain, since automatic information retrieval and machine learning techniques do not yet ensure a sufficient level of accuracy, even if some progress in the field has been made (for example in applying NLP techniques to recognize law modifications, as in [38]).

Manual markup of judicial decisions, however, doesn't seem to be sustainable in the long time. For an efficient management of the knowledge acquisition phase, a combination of tools supporting an authored translation of text into semantics should limit the effects of this still unavoidable bottleneck. Special editor tools (e.g. Norma-Editor) could enable an easy linking of text, metadata and ontology classes, while the more complex ontology constructs (e.g. the “considers/applies” chain) could be managed by an editor plug-in. Stronger constraints could be added to the legal core ontology in order to allow these plugins to automatically complete a part of the classification work, leaving to the user the duty of checking and completing the model drafted by the machine.

A possible new direction is to combine text fragment annotation à la Akoma Ntoso with semantic web machine reading as performed e.g. by


Explanation for Relevant_ExArt1341co2 SubClassOf Inefficacious

Axioms

◆Art1341co2cc applies Inefficacy_ExArt1341co2

●Inefficacious EquivalentTo Contractual_Agreement and (judged_as some Inefficacy)

◆Inefficacy_ExArt1341co2 Type Inefficacy

●Relevant_ExArt1341co2 EquivalentTo Contractual_Agreement and ((applies some Oppressive_Status) or (judged_as some Oppressive_Status)) and ((applies value General) or (judged_as value General)) and ((applies value NotSpecificallySigned) or (judged_as value NotSpecificallySigned)) and ((applies value Unilateral) or (judged_as value Unilateral))

●Relevant_ExArt1341co2 EquivalentTo considered_by value Art1341co2cc

▬considered_by o applies SubPropertyOf judged_as

Fig. 26. Explanation for Relevancy being inferred as a subclass of Inefficacious.

FRED[12] [55], which automatically produces a draft RDF/OWL ontology from a text, and links it to existing semantic web data. The supervision of an editor would then be much faster and easier to implement.

5.3.2. Representing exceptions

A critical issue in representing the decision's content is represented by *exceptions* to legal rules. How to model a situation when a material circumstance applies all the legal statuses required by the legal rule, but nevertheless does not fall under that legal rule's legal consequence because it follows some additional rule that defeats the first one? This issue has no straightforward solution inside the description logic fragment expressible in OWL2: once negative conditions are introduced for the rule to apply (in the form `if (not (exception))`), the open world assumption would prevent any inference on that rule, unless it is explicitly stated that no such exception exists. This would hinder the reasoning capabilities of classical OWL2 reasoners on the ontology library presented so far.

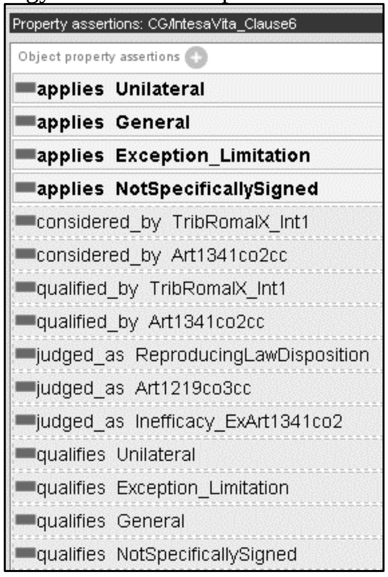
However, a solution to this problem might rely on modeling the exceptional case as a subclass of the normal case, (see Figure 25). In that way, only the instances that are relevant under the law are eligible to be an exception to the application of that law. This solution has the advantage of enabling reasoning on exceptions, without the need to apply defeasible rules. The backside is that the classification of the circumstance as exceptional is *added* to the classification of inefficacy, and not *substituted* to it (Figures 26 and 27).

Again, this issue originates from the open world assumption, and cannot be easily avoided while remaining inside OWL-DL: whenever the reasoner is prevented to link a circumstance to a legal consequence, asking him to check that no exception exists, the reasoner will be incapable of inferring anything unless all information concerning the exceptions is explicitly provided in the ontology.

This issue represents the main reason why a complete modelling of legal rules is not feasible within JudO, requiring instead a rule system (such as LKIF-Rules [24], Clojure, or LegalRuleML [41]) to be fully implemented.

6. Conclusions

We have presented an ontology design pattern, JudO, which enables representation and reasoning over the content of judicial decisions. The pattern uses the conceptualization behind the Descriptions and Situation framework, and a semiotic ontology, but also reuses (and is aligned to) the LKIF-Core legal ontology. JudO is then specialized in a sample


Property assertions: CG/IntesaVita_Clause6

Object property assertions

▬applies Unilateral

▬applies General

▬applies Exception_Limitation

▬applies NotSpecificallySigned

▬considered_by TribRomaIX_Int1

▬considered_by Art1341co2cc

▬qualified_by TribRomaIX_Int1

▬qualified_by Art1341co2cc

▬judged_as ReproducingLawDisposition

▬judged_as Art1219co3cc

▬judged_as Inefficacy_ExArt1341co2

▬qualifies Unilateral

▬qualifies Exception_Limitation

▬qualifies General

▬qualifies NotSpecificallySigned

Fig. 27. Stated / inferred property assertions on the exceptional contractual agreement.

[12] http://wit.istc.cnr.it/stlab-tools/fred

contract domain ontology, in order to show the capability of the pattern, and of some advanced features of OWL2 to express a larger part of legal knowledge in the OWL2 description logic fragment.

The ontology library presented in this article is the pivot of an innovative approach to case-law management, filling the gap between text, metadata, ontology representation and rules modeling, with the goal of detecting the information available in the text to be used to perform legal argumentation. This approach allows to directly annotate the text with peculiar metadata defined in core, domain and argument ontologies. OWL2 is used to get as close as possible to the rules, in order to exploit the computational advantages of description logics. This approach has a weakness in the management of exceptions: it is thus necessary to devolve this kind of reasoning features to different logics, e.g. defeasible logics, such as that presented in [28], with added support for argumentation schemes. Another limitation is that only text fragments get annotated, therefore deep semantic parsing is envisaged as a solution to this classical knowledge acquisition bottleneck.

References

[1] G. Antoniou, N. Dimaresis and G. Governatori, A Modal and Deontic Defeasible Reasoning System for Modelling Policies and Multi-Agent Systems, Expert Systems With Applications 36, 2, 2009, pp. 4125-4134.

[2] K. D. Ashley, Ontological requirements for analogical, teleological, and hypothetical legal reasoning, in: Proceedings of the 12th International Conference on Artificial Intelligence and Law, ACM, New York, 2009, pp. 1-10.

[3] J. L. Austin, How to do Things with Words, Second Edition, Oxford University Press, Oxford, 1975.

[4] G. Barabucci, L. Cervone, M. Palmirani, S. Peroni and F. Vitali, Multi-layer Markup and Ontological Structures in Akoma Ntoso, in: P. Casanovas, U. Pagallo, G. Sartor and G. Ajani, eds., AI Approaches to the Complexity of Legal Systems: Complex Systems, the Semantic Web, Ontologies, Argumentation, and Dialogue, Springer, 2010, pp. 133-149.

[5] T. Bench-Capon and T. F. Gordon, Isomorphism and argumentation, in: Proceedings of the 12th International Conference on Artificial Intelligence and Law, ACM, New York, 2009, pp. 11-20.

[6] E. Blomqvist, A. Gangemi and V. Presutti, Experiments in Pattern-based Ontology Design, in: Y. Gil, N. Noy, eds., Proceedings of the Fifth International Conference on Knowledge Capture, ACM, Los Angeles, 2009, pp. 41-48.

[7] F. Bobillo and U. Straccia, An OWL ontology for fuzzy OWL 2, in: J. Rauch, Z.W. Ras, P. Berka and T. Elomaa, eds., Foundations of intelligent systems, Springer, Berlin Heidelberg, 2009, pp. 151-160.

[8] F. Bobillo, M. Delgado and J. Gómez-Romero, DeLorean: A Reasoner for Fuzzy OWL 2, Expert Systems with Applications 39.1, 2012, pp. 258-272.

[9] G. Boella, G. Governatori, A. Rotolo and L. van der Torre, Lex minus dixit quam voluit, lex magis dixit quam voluit: A formal study on legal interpretation, in: P. Casanovas, U. Pagallo, G. Sartor, G. Ajani, eds., AI Approaches to the Complexity of Legal Systems. Complex Systems, the Semantic Web, Ontologies, Argumentation, and Dialogue, Springer, Berlin, 2010, pp. 162-183.

[10] A. Boer, R. Winkels and F. Vitali, Metalex XML and the Legal Knowledge Interchange Format, in: P. Casanovas, G. Sartor, N. Casellas and R. Rubino, eds., Computable Models of the Law, Springer, Heidelberg, 2008, pp. 21-41.

[11] S. Brüninghaus and K. D. Ashley, Generating legal arguments and predictions from case texts, in: G. Sartor, ed., Proceedings of the 10th international conference on Artificial intelligence and law, ACM, New York, 2005, pp. 65-74.

[12] P. Casanovas, M. Poblet, N. Casellas, J. Contreras, V. R. Benjamins and M. Blasquez, Supporting newly-appointed judges: A legal knowledge management case study, Journal of Knowledge Management, Emerald Group Publishing, Bingley, 2005, pp. 7-27.

[13] M. Ceci, Interpreting Judgements Using Knowledge Representation Methods and Computational Models of Argument, Ph. D dissertation, 2013, available at: http://amsdottorato.cib.unibo.it/6106/1/Marcello_Ceci_tesi.pdf.

[14] M. Ceci, Representing Judicial Argumentation in the Semantic Web, in: Proceedings of the V Workshop on Artificial Intelligence and the Complexity of Legal Systems (AICOL) within the 26th International Conference on Legal Knowledge and Information Systems, Springer, under publication.

[15] M. Ceci and T. F. Gordon, Browsing case law: An application of the Carneades Argumentation System, in: H. Aït-Kaci, Y. Hu, G. Nalepa, M. Palmirani, D. Roman, eds., Proceedings of the RuleML2012 Challenge, at the 6th International Symposium on Rules within the European Conference of Artificial Intelligence, vol. 874, 2012, pp. 79-95.

[16] M. Ceci and M. Palmirani, Ontology Framework for Judgement Modelling, in: M. Palmirani, U. Pagallo, P. Casanovas, G. Sartor, eds., AI Approaches to the Complexity of Legal Systems: Models and Ethical Challenges for Legal Systems, Legal language and Legal Ontologies, Argumentation and Software Agents, Lecture Notes in Computer Science vol. 7639, Springer, Berlin, 2012, pp. 116-130.

[17] S. Despres and S. Szulman, Construction of a Legal Ontology from a European Community Legislative text, in: T. Gordon, ed., Legal Knowledge and Information Systems. Jurix 2004: The Seventeenth Annual Conference, IOS Press, Amsterdam, 2004, pp. 79-88.

[18] A. Gangemi, Ontology Design Patterns for Semantic Web Content, in: Y. Gil, E. Motta, V. Benjamins, M. Musen, eds., The Semantic Web – 4th International Semantic Web Conference, Lecture Notes in Computer Science vol. 3729, Springer, Berlin, 2005, pp. 262-276.

[19] A. Gangemi, Design Patterns for Legal Ontology Construction, in: P. Casanovas, P. Noriega, D. Bourcier, F. Galindo, eds., Trends in legal Knowledge. The Semantic Web and the Regulation of Electronic Social

Systems, European Press Academic Publishing, Florence, 2007, pp. 171-191.

[20] A. Gangemi, Norms and plans as unification criteria for social collectives, in: Journal of Autonomous Agents and Multi-Agent Systems 16(3), 2008, pp. 70-112.

[21] A. Gangemi, Super-duper Schema: an OWL2+RIF DnS Pattern, in: V. Chaudry, ed., Proceedings of Deep Knowledge Representation Challenge Workshop at the Sixth International Conference on Knowledge Capture, AAAI Press, 2011.

[22] A. Gangemi, M. T. Sagri and D. Tiscornia, A Constructive Framework for Legal Ontologies, in: R. Benjamins, J. Breuker, P. Casanovas and A. Gangemi, eds., Law and the Semantic Web, Lecture Notes in Artificial Intelligence vol. 3369, Springer, Berlin, 2005, pp. 97-124.

[23] A. Gangemi, D. M. Pisanelli, G. Steve, A formal ontology Framework to Represent Norm Dynamics, in: R. Winkels, ed, Proceedings of the Second International Workshop on Legal Ontologies, University of Amsterdam, 2001.

[24] T. F. Gordon, Construting Legal Arguments with Rules in the Legal Knowledge Interchange Format, in: G. Sartor, N. Casellas, R. Rubino, eds., Computable Models of the Law: Languages, Dialogues, Games, Ontologies, Springer, Heidelberg, 2008, pp. 162-184.

[25] T. F. Gordon and D. Walton, The Carneades Argumentation Framework: Using Presumptions and Exceptions to Model Critical Questions, in: Proceedings of the First International Conference on Computational models of Argument, IOS Press, Amsterdam, 2006, pp. 195-207.

[26] T. F. Gordon and D. Walton, Legal Reasoning with Argumentation Schemes, in: Proceedings of the Twelfth International Conference on Artificial Intelligence and Law, New York, ACM Press, Amsterdam, 2009, pp. 137-146.

[27] T. F. Gordon, G. Governatori and A. Rotolo, Rules and Norms: Requirements for Rule Interchange Languages in the Legal Domain, in: A. Paschke, G. Governatori, J. Hall, eds., Rule Interchange and Applications, Berlin, Springer, 2009, pp. 282-296.

[28] G. Governatori and A. Rotolo, Defeasible Logic: Agency, Intention and Obligation, in: A. Lomuscio and D. Nute, eds., Deontic Logic in Computer Science, Springer, Berlin, 2004, pp. 114-128.

[29] M. Gruninger and M. Fox, The Role of Competency Questions in Enterprise Engineering, in: A. Rolstadås, Benchmarking – Theory and Practice, Chapman & Hall, pp. 22-31.

[30] R. Hoekstra, J. Breuker, M. Di Bello and A. Boer, LKIF Core: Principled Ontology Development for the Legal Domain, in: J. Breuker, ed., Law, Ontology and the Semantic Web, IOS Press, Amsterdam, 2009, pp. 21-52.

[31] E. Kontopoulos, N. Bassiliades, G. Governatori and G. Antoniou, A Modal defeasible Reasoner of Deontic Logic for the Semantic Web, in: International Journal on Semantc Web and Information Systems, 2011, pp. 18-43.

[32] W. Kusnierczyk, Nontological Engineering, in: B. Bennett and C. Fellbaum, eds., Proceedings of the 2006 conference on Formal Ontology in Information Systems, IOS Press, Amsterdam, 2006, pp. 39-50.

[33] H. P. Lam and G. Governatori, The Making of SPINdle, in: G. Governatori, J. Hall, A. Paschke, eds., Rule Interchange and Applications, Lecture Notes in Computer Science vol. 5858, Springer, 2009, pp. 315-322.

[34] D. T. Minh, T. Eiter and T. Krennwallner, Realizing Default Logic over Description Logic Knowledge Bases, in: C. Sossai and G. Chemello, eds., Symbolic and Quantitative Approaches to Reasoning with Uncertainty, Springer, Berlin, 2009, pp. 602-613.

[35] L. Mommers, Ontologies in the Legal Domain, in: R. Poli, J. Seibt, eds., Theory and Applications of Ontology: Philosophical Perspectives, Springer, 2010, pp. 265-276.

[36] D. Nute, Norms, Priorities, and Defeasible Logic, in: P. McNamara and H. Prakken, eds., Norms, Logics and Information Systems, IOS Press, Amsterdam, 1998, pp. 201-218.

[37] M. Palmirani and F. Benigni, Norma-system: A legal information system for managing time, in: C. Biagioli, E. Francesconi, G. Sartor, eds., Proceedings of the V Legislative XML Workshop, European Press Academic Publishing, Florence, 2007, pp. 205-224.

[38] M. Palmirani and R. Brighi, Model Regularity of Legal Language in Active Modifications, in: P. Casanovas, U. Pagallo, G. Sartor, G. Ajani, eds., AI Approaches to the Complexity of Legal Systems, Lecture Notes in Computer Science vol. 6237, Springer, Berlin, 2010, pp. 54-73.

[39] M. Palmirani, G. Contissa and R. Rubino, Fill the Gap in the legal Knowledge Modelling, in: G. Governatori, J. Hall, A. Paschke, eds., Rule Interchange and Applications, Lecture Notes in Computer Science vol. 5858, Berlin, Springer, 2009, pp. 305-314.

[40] M. Palmirani, M. Ceci, D. Radicioni and A. Mazzei, FrameNet model of the suspension of norms, in: K. Ashley, T. van Engers, Proceedings of the 13th International Conference on Artificial Intelligence and Law, ACM, New York, 2011, pp. 189-193.

[41] M. Palmirani, G. Governatori, A. Rotolo, S. Tabet, H. Boley and A. Paschke, LegalRuleML: XML-Based Rules and Norms, in: F. Olken, M. Palmirani, D. Sottara, eds., Rule – Based Modeling and Computing on the Semantic Web, Springer, Berlin, 2011, pp. 298-312.

[42] M. Palmirani, T. Ognibene and L. Cervone, Legal Rules, Text, and Ontologies over Time, in: H. Aït-Kaci, Y. Hu, G. Nalepa, M. Palmirani, D. Roman, eds., Proceedings of the RuleML2012 Challenge, at the 6th International Symposium on Rules within the European Conference of Artificial Intelligence, vol. 874, 2012, pp. 131-146.

[43] M. Palmirani, L. Cervone, O. Bujor and M. Chiappetta, RAWE: An Editor for Rule Markup of Legal Texts, in: P. Fodor, D. Roman, D. Arnicic, A. Wyner, M. Palmirani, D. Sottara, F. Levy, eds., Joint Proceedings of the 7th International Rule Challenge, the Special Track on Human Language Technology and the 3rd RuleML Doctoral Consortium, CEUR Workshop Proceedings, 2013.

[44] H. Prakken, Formalizing Ordinary legal Disputes: a Case Study, in: Artificial Intelligence and Law, 2008, pp. 333-359.

[45] Y. Ren, J. Z. Pan and Y. Zhao, Closed World Reasoning for OWL2 with NBox, in: Tsinghua Science & Technology, vol. 15, issue 6, 2010, pp. 692-701.

[46] E. Rosch, Prototype Classification and Logical Classification: The Two Systems, in E.K. Scholnick, ed., New Trends in Conceptual Representation: Challenges to Piaget’s Theory?, Lawrence Erlbaum Associates, Hillsdale, 1983, pp. 73–86.

[47] G. Sartor, Legal Concepts as Inferential Nodes and Ontological Categories, in: Artificial Intelligence and Law, 2009, pp. 217-251.

[48] I. Savvas and N. Bassiliades, A Process-Oriented Ontology-Based Knowledge Management System for Facilitating Operational Procedures in Public Administration, in: Expert Systems with Applications, 2009, pp. 4467-4478.

[49] J. R. Searle, Speech Acts: an Essay in the Philosophy of Language, Cambridge University Press, 1969.

[50] J. Shaheed, A. Yip and J. Cunningham, A Top-Level Language-Biased Legal Ontology, in: J. Lehmann, M. A. Biasiotti, E. Francesconi, aM. T. Sagri, Workshop Proceedings on Legal Ontologies and Artificial Intelligence Techniques, Workshop Series No 4, Wolf Legal Publishers, Bologna, 2005.

[51] E. Sirin and B. Parsia, SPARQL-DL: SPARQL Query for OWL-DL, in: C. Golbreich, A. Kalyanpur, B. Parsia, eds., Proceedings of the OWLED 2007 Workshop on OWL: Experiences and Directions, CEUR-WS, 2007.

[52] F. Vitali, Akoma Ntoso Release Notes, 2011, available online at www.akomantoso.org/release-notes.

[53] W3C Consortium, OWL 2 Web Ontology Language Overview, 11 December 2012, available online at www.w3.org/TR/2012/REC-owl2-overview-20121211/.

[54] D. Walton, C. Reed and F. Macagno, Argumentation Schemes, Cambridge University Press, Cambridge, 2008.