An OWL ontology library representing judicial interpretations

Marcello Ceci[a*] and Aldo Gangemi[b,c]
[a] *CIRSFID, University of Bologna, Via Galliera 3, 40141 Bologna, Italy*
[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 a formal model of legal knowledge that relies on the metadata contained in judicial documents, and JudO, a judicial ontology library that represents the interpretations performed by a judge when conducting legal reasoning towards the adjudication of a case. For the purposes of this application, judicial interpretation is intended in the restricted sense of the acts of judicial subsumption performed by a judge when considering a material instance (a token in Searle's terminology), and assigning it to an abstract category (type). JudO is centred on a core ontology featuring some judicial ontology patterns, which take advantage of constructs introduced by OWL2, in order to provide appropriate legal semantics, while retaining a strong connection to source documents (i.e. fragments of legal texts). The final goal of the framework is to detect and model of jurisprudence-related information directly from the text, and to perform shallow reasoning on the resulting knowledge base. JudO also constitutes the basis for the application of argumentation patterns through reasoning on a set of rules, which represent the grounding of judicial interpretations in deontic and defeasible logics.

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

«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 behaviours, judicial activity enforces law provisions within national borders, therefore 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, but they rather codify 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

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

the combination of its 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 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. of other norms, or judgements). 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 [24,26,43].

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 [37]. 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 the previous efforts of the community in the field of legal knowledge

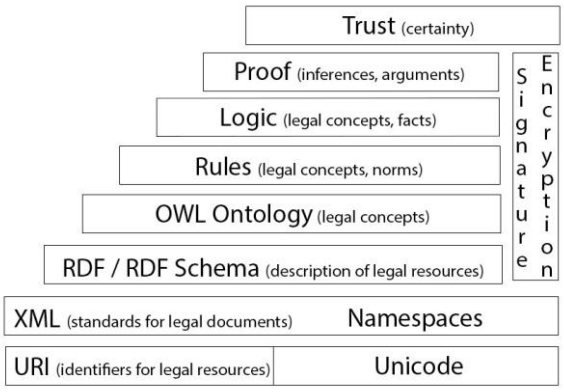

Fig. 1. Tim Berners Lee's Semantic Web layer cake, adapted to the legal domain in [46].

representation [34] and rule interchange for applications in the legal domain [26]. The issue of implementing logics to represent judicial interpretation has already been faced e.g. in [9,22], albeit only for the purposes of sample cases. The aim of the research is to apply legal theories to a set of real legal documents, possibly defining OWL axioms in a *Judicial Ontology Library* (JudO) that provides a semantically expressive representation, and a solid ground for a (future) argumentation system that applies a defeasible subset of predicate logics. The JudO ontology library will be the cornerstone for a semantic tool that is able to deepen, enrich, and reason on the XML mark-up of precedents (i.e. the metadata used for annotating case-law), and supports legal reasoning in the large.

Some new features in the more recent version of OWL (OWL2, see [52]) 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 main legal concepts in a specific domain concerned by case-law (e.g. contracts, e-commerce services, tort law, etc.);
d. An **argumentation system** [15], modeling the structure of argumentation (arguments, counterarguments, premises, conclusions, rebuttals, proof standards, argument schemes, etc.).

The present paper introduces the issues related with the *core* and *domain ontologies* – points b. and c. – which have been designed to organize the metadata annotating the text of judicial decisions, and to 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 [41]) and SPINdle (see [42]).

The paper is structured as follows: section 2 presents the requirements and the methods for the design of the ontology library; section 3 describes

how the ontology library is built, and how it manages knowledge related to judicial interpretation. In section 4, the method is exemplified with reference to a sample of Italian case law. Section 5 presents an evaluation of the ontology, discussing related work in either legal ontology or legal reasoning fields, and some issues of the proposed solution.

2. Tasks and applications

The research described in the present paper aims at applying state-of-the-art techniques in ontology design and DL reasoning to knowledge 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 using 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 [27] seem 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.

Moreover, argumentation theories (including the dialogue model of adjudication by [43], and argumentation schemes by [25]) 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 often 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 participate in reasoning, although providing less inferences. In addition, some relevant domain concepts in law can be designed with class axioms instead of rules, so providing an explicit account of domain-level classical reasoning. The `Relevant_Ex<rulename>` classes in JudO, under which all instances relevant to a specific law are automatically classified (see 3.3.1.), is an example of such design choice.

The target of the ontologies presented in this paper is that subset of legal knowledge, in order to enrich the metadata annotating a legal document by performing deductive reasoning on a knowledge base, and thus preparing it 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, application requirements. Functional ones 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 [38]. This means that the ontology should be easily extended with entities extracted from text, therefore it should contain as many constraints as needed for judicial reasoning, without over constraining 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), 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 a 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 laws.
- **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 may overrule the first);
  * *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. It is a sort of framing: in modeling those links, we need in fact 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[22].

- **Querying**: being able to perform complex querying, e.g. by using SPARQL-DL [50], 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 the knowledge extracted from the decision text can be processed and managed, and reasoning on the judicial interpretation grounding the decision is made possible. Reasoning intends to satisfy the following domain requirements[3] (also known as *competency questions*, see [28]):

- **Finding relevant precedents** that are not explicitly cited in the decision. In order to 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 let them emerge indirect connections between concepts. Certainly, in order to take the most

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

[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.

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, e.g. by using a plugin to a word processor that employs NLP techniques to recognize sentences and clauses that could be relevant under e.g. consumer law;
- **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 (e.g. integrated into a word processor to assist judges while writing judgements, so avoiding grounds for appeals due to missing elements in the decision's groundings);
- **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, e.g. by using a graphical interface and a formal argumentation structure instead of plain text.

3. Ontology Design

The approach adopted by the present research is based on a multi-layer paradigm, where a legal resource is managed in separate levels that are linked to each other, and organized in order to allow multiple annotation, interpretation, and classification with representation redundancy. The syntactical approach is based on the following schema:

- **Text annotation in XML**: the Akoma Ntoso standard [4,51] 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 the legal qualification of relevant parts of the decision, such as the minority report or the dissenting opinion;
- **Ontology annotation**: external OWL definitions linked to the XML document are used;
- **Rules**: unfortunately OWL, even with the functionalities of version 2.0, 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 in two modules (see also [16]):

- A **Core Ontology** describing the constituents of a precedent in terms of general concepts, through an extension to the LKIF-Core legal ontology;
- a **Domain Ontology** representing the concepts and the rules expressed by the Italian *Codice del Consumo* (Consumer Code) and in *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: bottom-up for capturing and modeling legal domain ontologies, and top-down for modeling core ontology classes and argumentation theory components. Middle-out methodology is implemented here by using pattern-based design [6,19] with Ontology Design Patterns either extracted from judicial text or reused from the core ontology, and matched according to requirements.

In order to manage the content of judgements, it is necessary to introduce particular structures to represent the instantiation of legal figures, such as *judicial interpretations*, which involve:

- **acts of interpretation**, which take into consideration a fact and apply a legal rule (legal status) 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).

The abstract categories of *qualifying expressions* (see 3.2.1.) are aimed at capturing this layered stack

of interpretations, 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 decisive.

Evaluation has been performed on a sample set of Italian case law including 27 decisions of different grade (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 the highlight of single clauses and their comparison to general rules as well as to case law concerning the matter. These possibilities can be used to introduce a semi-automatic compliance check of a contract draft. The domain considered is also interesting as it involves situations where strictly deductive logic is not sufficient to represent the legal reasoning as performed by a judge. In particular, defeasible logics [27] seem 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 help designing the background for applying defeasible logic is therefore an important goal 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 particular type of document modeled to detect the relevant parts of 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 detecting 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

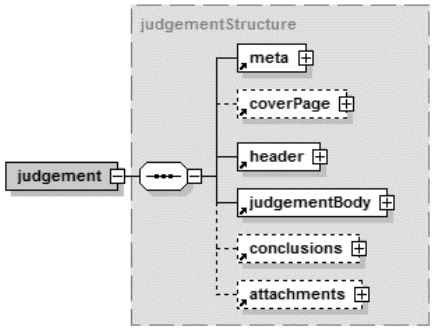

Fig. 2. Judgement structure in Akoma Ntoso.

supporting his decision; *decision*, where the final outcome is given by the judge.

This partition allows detecting facts and factors from the background: in the motivation part, arguments and counterarguments are detected, while in the decision part lies the conclusion of the legal argumentation process. Those qualified fragments of text should be annotated by legal experts with the help of a special editor (e.g. Norma-Editor, presented in [36]) that is handy to create links between text, metadata and ontology classes.

3.2. Core Ontology

The judicial core ontology[4] (Figure 3) introduces the main concepts in that legal domain, defining the

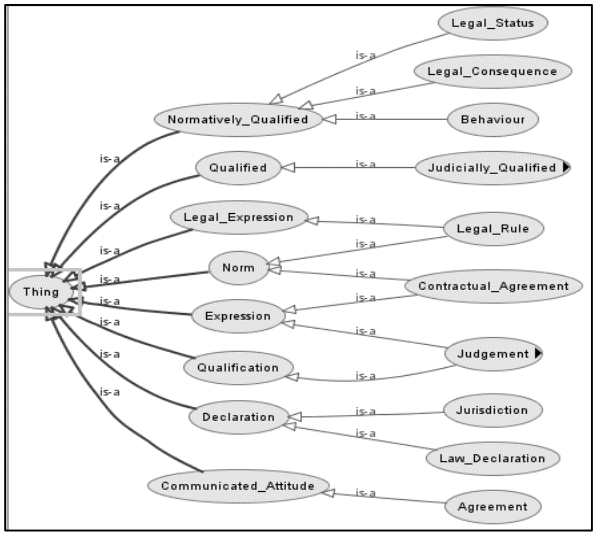

Fig. 3. Core Ontology's specification of LKIF-Core. The central column defines categories already present in LKIF-Core, whose further classification is not contained in the graph.

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

classes that include entities extracted from judicial decisions. Core ontologies are domain-generic and not modeled upon a specific legal subject, however being the legal domain too large and heterogeneous, the model presented here is conceived to represent interactions in Civil Law, especially as far as 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

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

`Qualifying_Legal_Expression` 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 `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 intellectu al work", and "*x* is a technical invention" are *qualifications*. Consider that legal acts are typically *speech acts* (cf. [3]) that influence the behavior of people and institutions by means of the performative or normative value of the meaning expressed in those acts (semiotics.owl[5] is an ontology design pattern formalizing speech acts in OWL, cf. also [19] for an application to legal ontologies) . The modeling of qualifying legal expressions also takes into consideration Searle's theory of constitutive acts and distinction between *fact-tokens* and *fact-types* (see [48]).

The `Qualified` class includes anything that is object of a qualification. In the examples, both "*x*" (e.g. a *material circumstance* i.e. a *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* [31]. In cognitive science, this means that qualifications introduce both a *categorization*, and a *prototype* [45].

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.

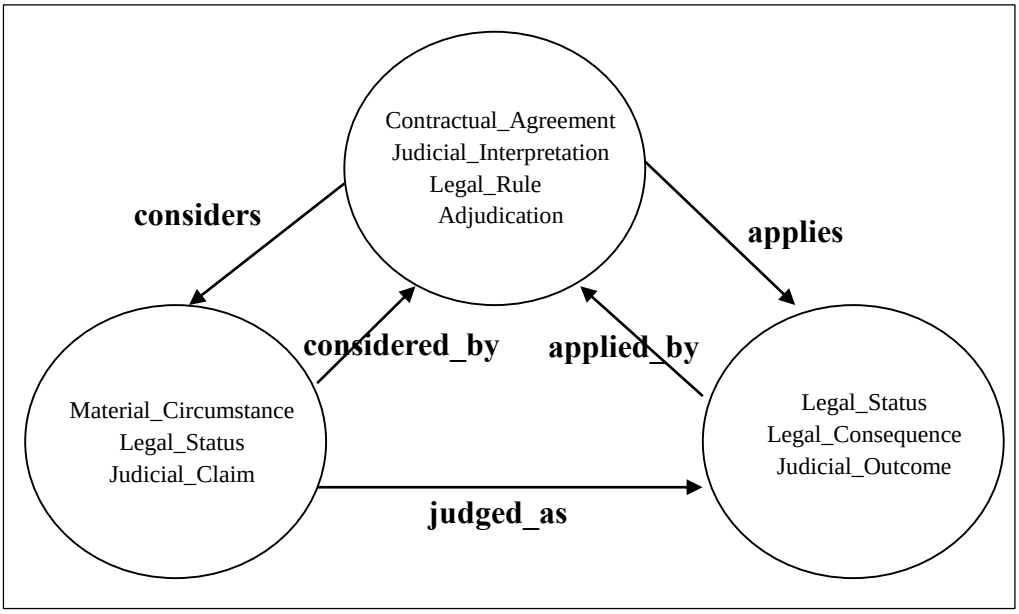

Fig. 4. Interaction between qualifications, tokens and types.

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

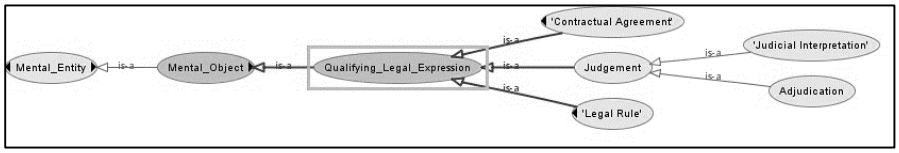

Fig. 5. Visualization of the Qualifying Legal Expression class.

The three classes are represented in an ontology design pattern [19], which specializes a part of the Description and Situations pattern [18,20]: qualifications are a subclass of descriptions (expressed by qualified legal expressions) that characterize qualified elements (either at the instance and type levels), and that 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` respectively). 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 **qualifi es**(*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 also 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, e.g.:

- 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 (See [29]) is an established legal ontology, and we want to be compatible to it. In this section we explain some of the measures taken to obtain this compatibility. We also reuse some of its classes and properties in JudO when the concepts represented in LKIF fulfill JudO requirements.

JudO's `Qualifying_Legal_Expression` class (Figure 5) is aligned to the union of the `lkif:Legal_Expression` (Figure 6) and `lkif:Qualification` (Figure 7), enhanced by the specialization of the `lkif:qualifies` property into `considers` (modeled as a superclass of the LKIF-Core properties `evaluates`, `allows`, `disallows`) and `applies`.

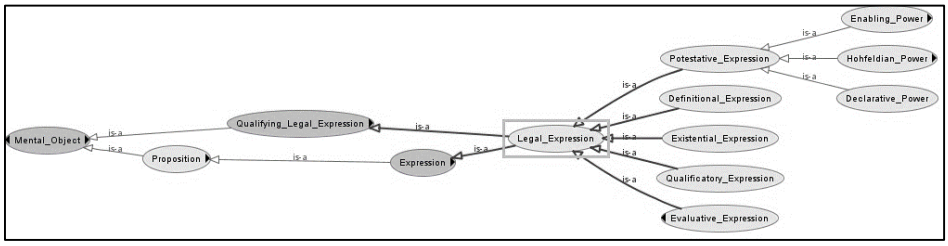

Fig. 6. Visualization of the Legal_Expression class.

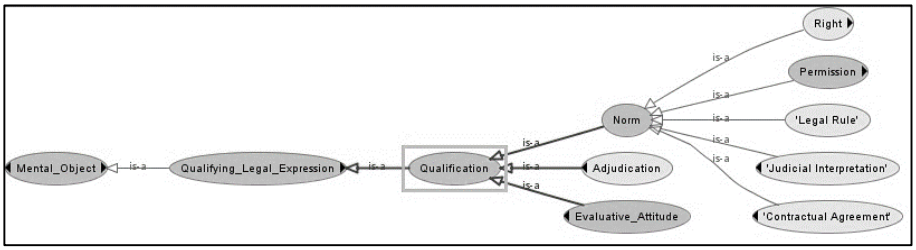

Fig. 7. Visualization of the Qualification class.

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`.

As a superclass of `lkif:Legal_Expression` (Figure 6), instances of `Qualifying_Legal_Expression` 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 (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 `Medium`, `Attitude` and `Agent` classes). As a superclass of `lkif:Qualification` (Figure 7), `Qualifying_Legal_Expression` 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:Qualified` classes (the latter representing both qualifying –type-level– and qualified things –instance-level) are linked only by a single property (`lkif:qualifies/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 respectively the *object* (instance-level) and the *destination* (type-level) of the qualification.

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`.

`Normatively_Qualified` expressions include instances of `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 the `Norm`, `Legal_Status` represents an institutional fact (i.e. 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 (or denied) 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.

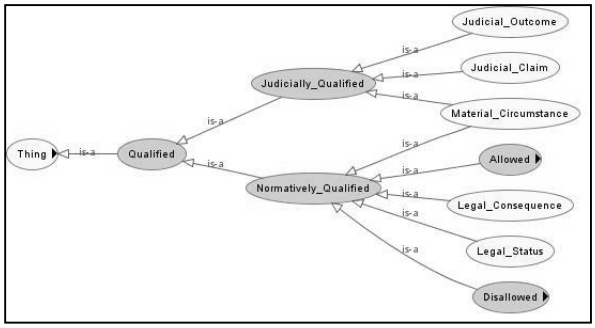

Fig. 8. Visualization of the Qualified class.

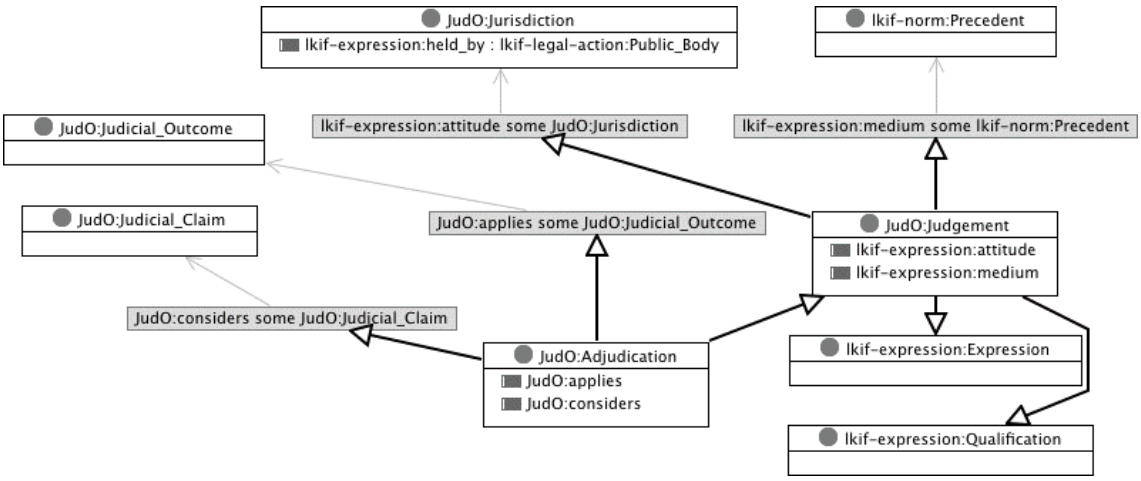

Fig. 9. Visualization of the adjudication class and of its semantic connections.

`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`, expecially 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 `Qualification>Judgement`). 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`. So 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`, expecially 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 `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_Interpretation`. For example, a `Contractual_Agreement` can be `considered_by` some `Judicial_Interpretation` that `applies` some `Legal_Status` t o it (e.g. the agreement can be *oppressive*, *inefficacious*, can *represent an arbitration clause*, can be *specifically signed by both parties*). In these cases, an OWL2 property chain directly links a `Contractual_Agreement` to the `Legal_Status` judicially applied to it. This property, called `judO:judged_as`, enriches the judicial qualification ontology design pattern presented above. See 3.4.1. for a description of this feature and of its usage in the present researchMedia, Propositional Attitudes and Agents

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

The `Medium` cla ss identifies the support, through which a proposition is expressed. In JudO, the `medium` property has not been used to represent the material support of an `Expression`, but rather its

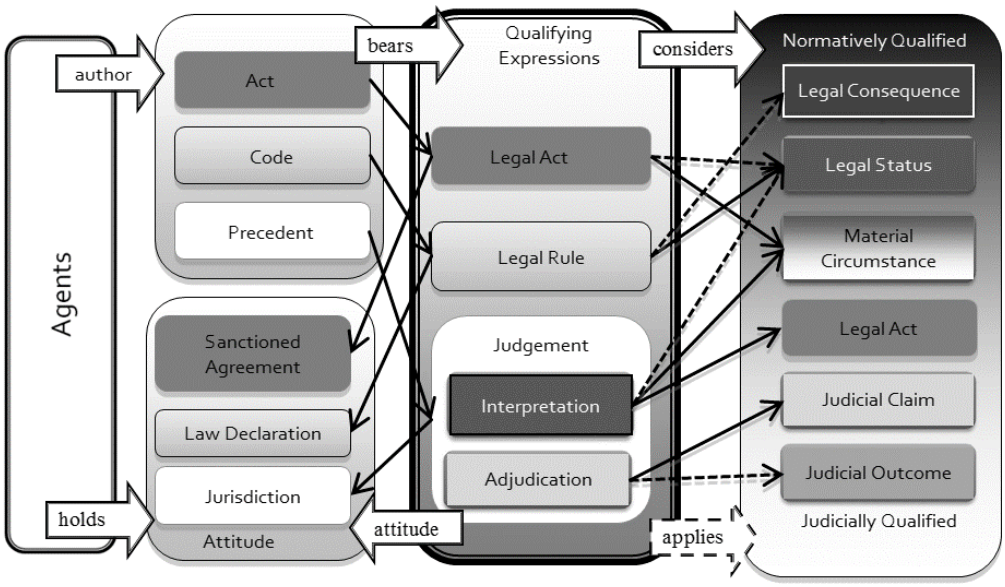

Fig. 10. The Core Ontology graph. Boxes represent classes. Continuous arrows represent either the bears, attitude or considers properties. Dashed lines represent the applies property.

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 representing the enabling powers behind a `judo:Judgement`, a `judo:Legal_Rule` or a `judo:Contractual_Agreement` respectively. On the contrary, 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

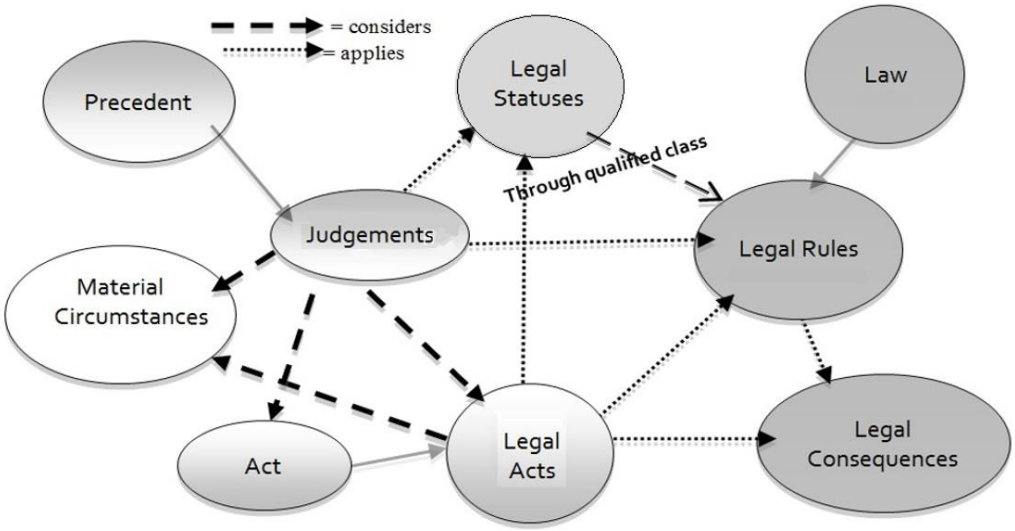

Fig. 11 – semantic relations between represented knowledge. The dashed line "Through qualified class" means that the connection from legal statuses to legal rules is ensured through a qualified class (see 3.3.1.).

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 in representing critical factors of particular precedents.

3.2.5. 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 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 9 repre sents the classes and properties of the core ontology. Figure 10 shows the same information, but allows to better understand the connection 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 an expert in the legal subject, which actually represents the only viable choice in the legal domain, albeit giving rise to important bottleneck issues (see below 5.3.1.).

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

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

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

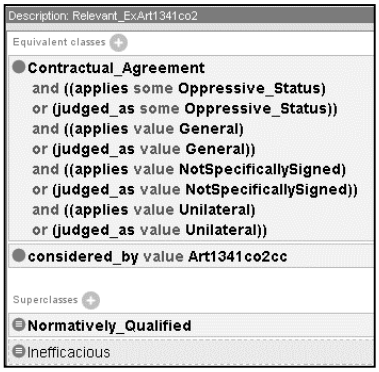

Fig. 14. Axiom for the classification of Contractual Agreements under the legal rule Art. 1341 comma 2.

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 modelling of the same concept by different authors).

3.3.1. Modelling of laws

The laws involved in the 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. Legal rules are also represented through anonymous subclasses of the `Normatively_Qualified` class (Figure 13), according to the template `Relevant_Ex<rulename>` (*ex* is the latin proposition for indicating a source). An axiom stating the requirements for an instance to be relevant under the legal rule is included in the description of the class, as well as an equivalence 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 anonymous classes are classified under the `Contractual_Agreement` class: that is, 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.

3.3.2. Modeling of contracts

A contract is a composition of one or more `Contractual_Agreements` (a `Contract` for the whole, 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 specific interpretations, capturing tiny factors that led to that interpretation.

3.3.3. Modelling of judicial decisions

The instances of the `Judgement` class include an instance identifying the case as a whole (the precedent) and several others identifying 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 (claimant, defendant, and court). An `Adjudication` contains the `Judicial_Outcome` of the `Judicial_Claim`.

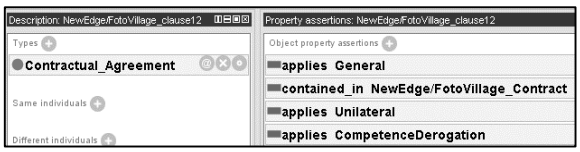

Fig. 15. Description and property assertions of the contract clause's content.

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

(it considers the claim and applies the outcome), while a Judicial_Interpretation considers a Material_Circumstance and applies one or more Legal_Status (and zero or more Precedents) to it. The precedents cited by the judge in the decision are added directly to the interpretation instance: the reasoner is then capable of distinguishing between legal statuses and precedents, the latter being searchable in queries and other information retrieval applications. Rules expressed by precedents (i.e., if a clause is signed through a recall at the end of the document, it is specifically signed) can be modeled in the same way 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[7] reasoner. This tool was built to extract data from the OWL ontology, but could also be used to check if the ontology gives a unique and correct answer to some formalized question (i.e. 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 on the documents involved: contract parties, judicial actors, legal status applied to the agreement (eventually in comparison to the one suggested by the contract/judicial parties), the rules of law which are relevant to the legal status, the final adjudication of the claim, the part played in it by the interpreted agreement, and so on.

The most 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 the contract and the applicable law. The ontology reasoner is in fact capable of predicting – to some extent – the outcome of the judge (i.e. predicting that a clause will be judged as valid/invalid) and to run inferences about the agreement (for example, 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).

[7] 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. OWL2 Constructs Used

OWL2 (see [52]) is one of the latest standard for the Semantic Web, and is relevant to any project willing to contribute to the huge network of data that is being built on the Web (a large part of which is now called "Web of Data"). An objective of the present research is to explore how OWL2 could help designing the background for the application of defeasible logic: OWL, in fact, is not designed for managing defeasibility directly, being only able to capture the static factual and legal 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. For this reason, well aware of the limitations of OWL2 in representing defeasible logics, one aim of the present research is to investigate how far OWL2 can be used in order to

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

improve performance, computability, and management of classes in a defeasible logic context.

OWL2 introduces several features to the original Web Ontology Language, some of which allow a richer representation of knowledge, especially when dealing with properties and datatypes. Some of these would be useful, but also lead to a great increase of complexity in the models: for example, in order to exploit disjointness between properties, it would be necessary to create as many properties as possible statuses, which in turn would greatly affect the intricacy and readability of the ontology. On the contrary, some of these new constructs concerning properties deserve attention because they could enhance expressivity without affecting (or even reducing) the complexity of the model built so far. OWL functional syntax will be used in examples throughout the paper.

3.4.1. Property Chains

The OWL2 construct `ObjectPropertyChain` used within a `SubObjectPropertyOf` axiom allows a property to be defined as the composition of several properties as in Figure 18. Such axioms are known as *complex role inclusions* in SROIQ. JudO relies on one particular property chain useful in the judicial domain. The property chain:

```
considered_by o applies SubObjectPropertyOf judged_as
```

is represented in Figure 19, and is used in two different ways – in interpretations, as in the figure, and in rule applications – to create a direct interpretational link between a *material circumstance* and a *legal status*.

When a `Judicial_Interpretation considers` a `Material_Circumstance` and `applies` a `Legal_Status`, the `judged_as` property chain comes into play and creates a direct link between the circumstance and its status, that link being distinguished from the indirect one introduced by the contract (represented by the property applies). Reasoners will therefore treat these two links accordingly.

On the other hand, the 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`).

A better use of the OWL2 property chains could lead to an ever more direct and complete solution, mainly 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, in fact, the property chain `judged_as` connects a material instance (i.e. contract clause) to a legal status or legal consequence (i.e. `oppressive`, `inefficacious`) via a judicial interpretation. With the open world approach, this creates a sprawling of `judged_as` chains being applied to the metadata. All of these inferences are correct; nevertheless, they greatly increase the number of triples in the ontology. In order for the ontology to manage a big knowledge base and to perform deep reasoning on it, 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.

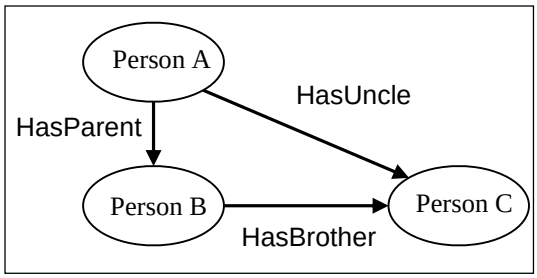

Fig. 18. An example of property chain.

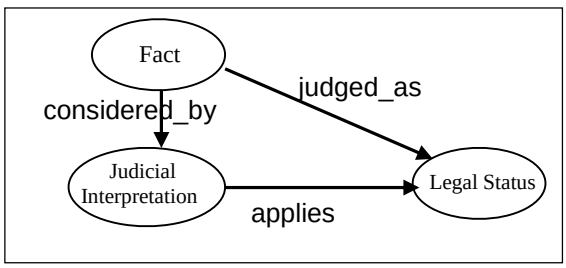

Fig. 19. The property chain judged_as.

3.4.2. *Negative object properties*

A negative object property assertion such as:

```
NegativeObjectPropertyAssertion(O
P a b)
```

states that the individual `a1` is not connected by the object property `OP` to the individual `a2`. E.g. given an ontology including the following axiom:

```
NegativeObjectPropertyAssertion(h
asSon Peter Meg )
```

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

```
ObjectPropertyAssertion(hasSon
Peter Meg)
```

Negative object property assertions are useful to avoid complicated workarounds for negating assertions. For example, the legal status `NotSpecificallySigned` and more constructs are needed in OWL1 in order to represent the statement that a certain status is not SpecificallySigned, e.g.:

```
EquivalentClasses(SpecificallySig
ned?
ObjectOneOf(NotSpecificallySigned
SpecificallySigned))

DifferentIndividuals(Specifically
Signed NotSpecificallySigned)

ObjectPropertyAssertion(applies
ContractA NotSpecificallySigned)
```

but in OWL2 the following construct is sufficient:

```
NegativeObjectPropertyAssertion(a
pplies                    ContractA
SpecificallySigned)
```

3.4.3. *Key*s

A `HasKey` axiom states that each *named* instance of a class is uniquely identified by a set of data or object properties assertions - 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 for identifying the unique elements in a judicial claim, e.g. the parties, the contract, the norm, and the decision itself.

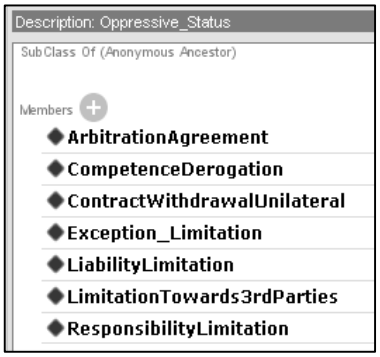

Fig. 20. The list of legal statuses classified as oppressive.

3.4.4. *Annotation properties*

OWL1 allows extra-logical annotations to be added to ontology entities, but does 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.

3.4.5. *N-ary datatypes*

In OWL it is 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 note proposes an extension to OWL2 for defining data ranges in terms of linear (in)equations with rational coefficients. This kind of equations is of high importance in the process of identifying individuals to classify under a legal ontology framework on the basis of a quantitative evaluation of the relationship between several factors.

3.4.6. Property qualified cardinality restrictions

While OWL1 allows for restrictions on the number of instances of a property (i.e. for defining persons that have at least three 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, and ObjectExactCardinality (respectively DataMinCardinality, DataMaxCardinality, and DataExactCardinality). These restrictions, together with n-ary datatypes, are fundamental 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 applciation 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[8], 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.

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 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.

The disposition is represented as a Qualifying Legal Expression (`Legal_Rule`) called "art1341Co2" (with a `Code` medium, a `Law_Declaration` attitude and a `Parliament` as agent) and the qualified class `Relevant_ExArt1341co2`. As seen in 3.3.1, a `Legal_Rule considers` a (combination of) `Legal_Status`(es) and `applies` a `Legal_Consequence` (or a deontic operator). Therefore any individual which 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 20) will be `considered_by` "art1341Co2", which in turn `applies` the `Legal_Consequence` of "invalidityExArt1341co2". The individuals "competentJudge" and "notSpecificallySigned" are thus created as `Legal_Status`es that can be

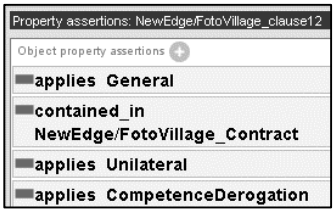

Fig. 21. Stated property assertions for the sample agreement.

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

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 21) is created and linked to a Contractual_Agreement which applies the Legal_Statuses of "General", "Unilateral" and "CompetenceDerogation". 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.

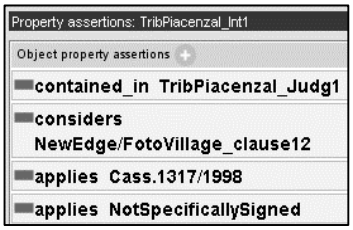

Fig. 22. Stated property assertions of the sample judicial interpretation.

4.3. Modelling of the judicial interpretation

The Judgement instance is created, as well as its components (single interpretation instances, adjudication, etc.). Among them, the "tribPiacenzaI_Int1" Judicial_Interpretation is created (Figure 22): it conside rs the Contractual_Agreement contained in "α/βClause12", and applies the "notSpecificallySigned" Legal_Status. The instance contains also a reference to the precedent

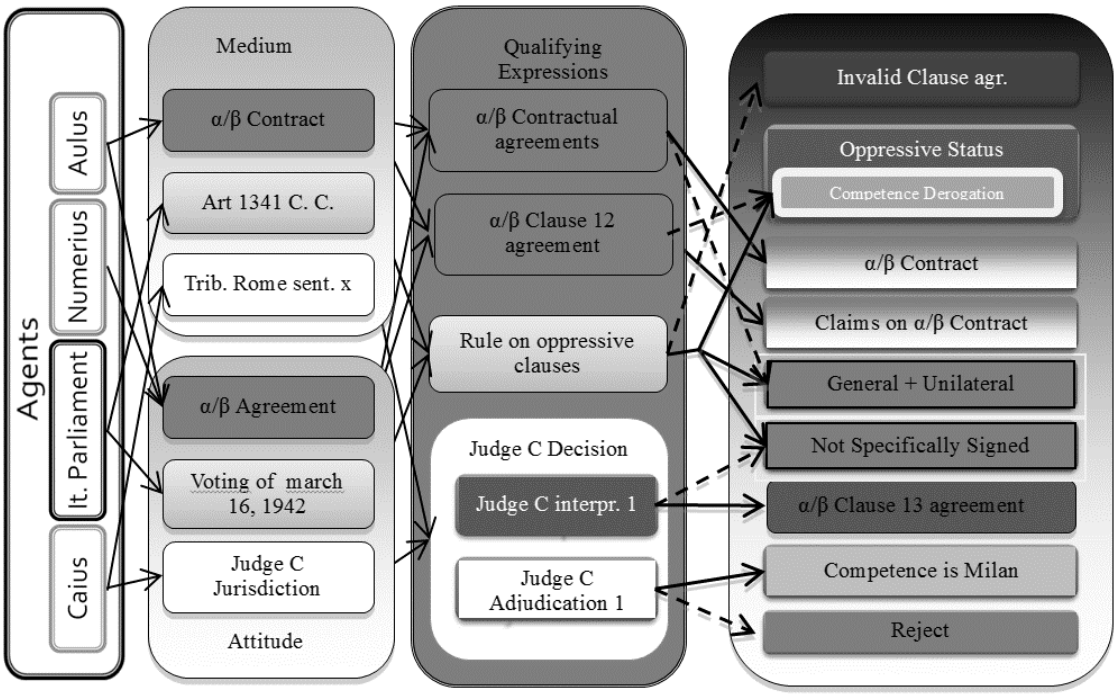

Fig. 23. The graph showing the model of the sample case. The general classes of fig. 11 have been substituted with the sample instances. The properties (arrows) connect the same classes of the core ontology.

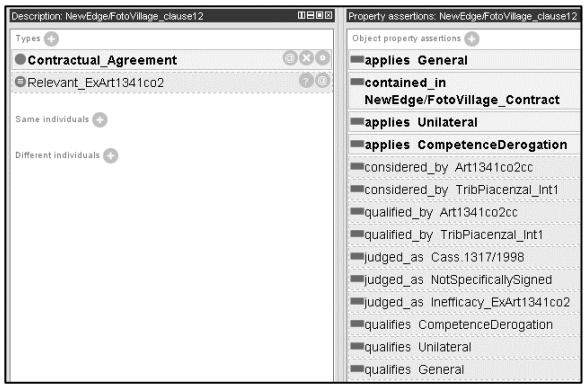

Fig. 24. Inferred Description and property assertions of the contract clause's content.

(Cass.1317/1998), which represent a semantically-searchable information on the interpretation instance. Figure 23 shows 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, the 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 24). 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 fulfill the axiom are automatically classified in that class, and thus `considered_by` the proper law. At this point, the `judged_as` property chain links the clause to the legal consequence through the legal rule (clause is `considered_by` the law which `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 25 explains the whole process as a list of axioms verified by the ontology reasoner.

5. Evaluation of the ontology library

The ontology library, in its sample taken from real judicial decisions, met 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 of text and of the legal concepts they contain.
- **Distinction between document layers**: The qualifying expression class constitutes the main expressive element, introducing an n-ary relation that ignites the reasoning engine. Its instances can refer to the same text fragment, yet represent different (and potentially inconsistent) interpretations of that text.

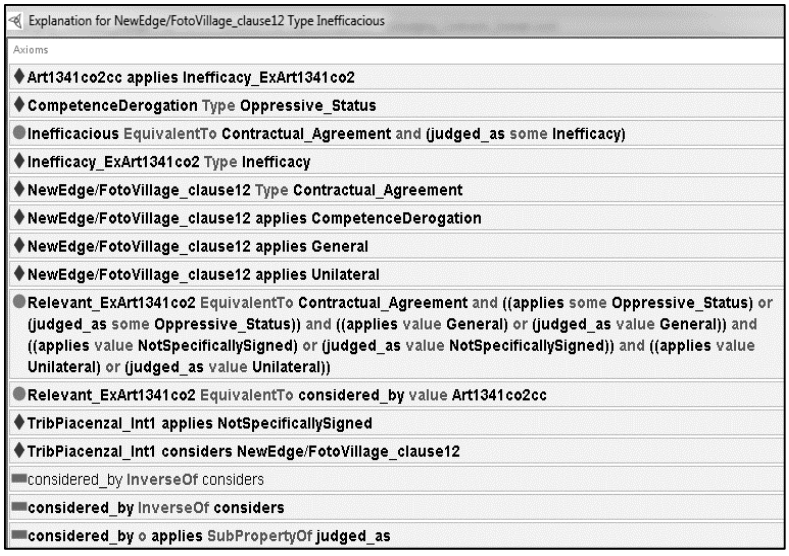

Fig. 25. Explanation for the sample agreement being inefficacious.

Moreover, the LKIF-Core's `Medium` class allows to represent 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, also with the purpose of easening the effort needed to model all knowledge contained in the ontology. These axioms could also be used to support tools that automatically complete partially-modeled documents;
- **Querying**: the considers/applies properties allow complex querying on the knowledge base, and the judged_as shortcuts provide semantic sugar in this perspective. Querying on temporal parameters is not yet possible due to limits in LKIF-Core language: solutions for this are being achieved through emerging standards for rules such as LegalRuleML.
- **Modularity***:* the layered (core/domain) structure of the ontology library renders domain ontologies independent between each other - and yet consistent, through their compliance to the core ontology template.
- **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 a *lite* 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 on 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 Chapter 4). This could be improved by optimizing the reasoner and/or with a further refinement of the ontology (and rules) structure.

5.1. 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 [26]. 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.

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 the legal operations carried out by the judge in a decision, mainly judicial interpretations seen as subsumption of material facts or circumstances under abstract legal categories.

The project presented in [47] 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 [47] 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] shows an 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 organized 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 [49] 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 ([49] contains a comparison to LRI-Core), and relies on agents to bridge the legal text with the syntax. The approach is very interesting, yet the focus on agents somewhat 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 the reasoning on a large scale of document contents (for a general account on how to model complex legal concepts for automatic detection see [39]). 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] allows combining the features of the DL-based ontology with non-monotonic logics such as Defeasible Logics. In particular, Carneades is based on Walton's theory [25] and also gives account for most of Prakken's consideration on the subject [43] 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 Prof. Kevin Ashley such as [2]. Their main focus is similar to the one of the present research: capturing the elements that contribute to the decision of the judge. The approach was, however, based on concepts rather than on the legal documents themselves. They were meant to teach legal argumentation in law classes. No account for the metadata of the original text was given, and there was 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 (under criteria of time, hierarchy, and others). Of course, *templatizing* legal documents is a very complex task (see next section, 3.5.1.): the intention, in any case, is not to provide a complete NLP tool but to create an interface through which a legal expert can easily identify the legal concepts evoked by single words, and combinations of them, in legal documents.

Deontic defeasible logic systems, such as those presented in [27,30,35] 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 contained in ontologies such as the one presented in this paper. These projects are therefore placed at an upper layer than the one discussed here: the ontology, in the perspective of the present research, should focus on the document semantics and basic relations, in order to perform shallow reasoning oriented mostly to data completion, enhanced by the open world assumption. Over a such-built knowledge base, rule systems based on advanced logic dialects (such as those presented in the cited works) could perform complex reasoning with tools such as SPINdle (see [32]) by importing only the set of triples that best suits their syntactic 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 shallow reasoning while staying within OWL2, and in order to perform an efficient reasoning on legal concepts it is not sufficient to implement defeasible reasoning, being also necessary to rely on argumentation schemes [53]

The same considerations apply to the approach in [33], which interestingly provides a simple and intuitive way to encode default knowledge on top of terminological KBs: such a reasoning system does not reach the complexity needed to manage legal concepts (for which deontic defeasible logics are required, with an account for argumentation schemes). This means that a distinct layer is needed in order to perform deep reasoning on the KB: being this the situation, it is better to stay within the achieved standard of OWL2 when performing basic reasoning on KB.

In this perspective, the idea of deriving a closed-world subset of an OWL2 KB as presented in [44] seems an optimal enhancement of the present ontology, and will in fact be explored, always 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.

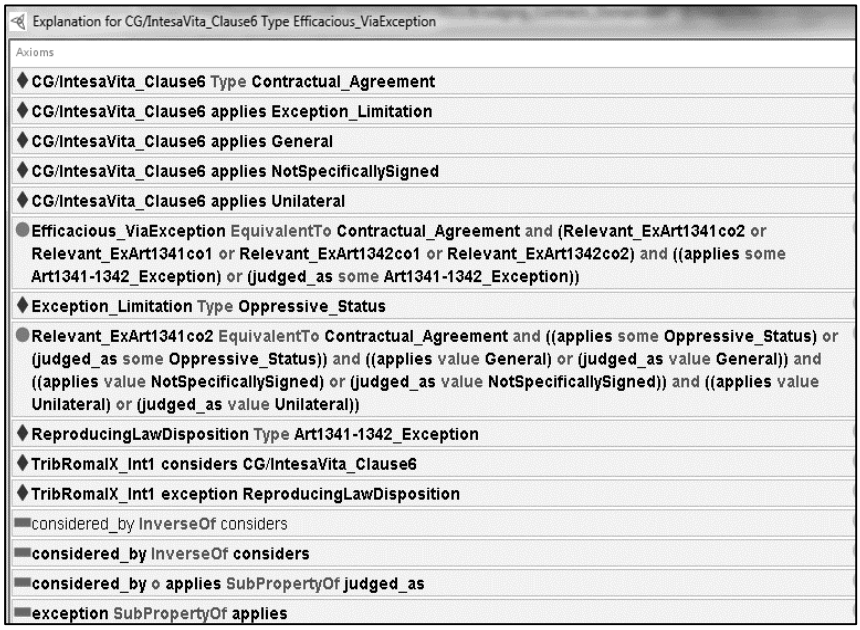

Fig. 26. Explanation of a sample contract clause being not inefficacious because of an exception.

5.3. Issues

5.3.1. The knowledge acquisition bottleneck

The modelling of the sample ontology library and the extraction of knowledge from the case law sample was carried out manually by a graduated jurist. Also the qualified fragment of text under the Akoma Ntoso standard are supposed to be annotated by legal experts: at the present time, manual data insertion seems the only viable choice in the legal domain. In fact, 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 [37]).

The 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 between text, metadata and ontology classes, while the more complex ontology constructs (i.e. the "considers/applies" constructs) could be managed by an editor plug-in. In this perspective, stronger constraints could be added to the legal core ontology in order to allow these plugin to automatically complete a part of the classification work, leaving to the user the duties of checking and completing the model drafted by the machine.

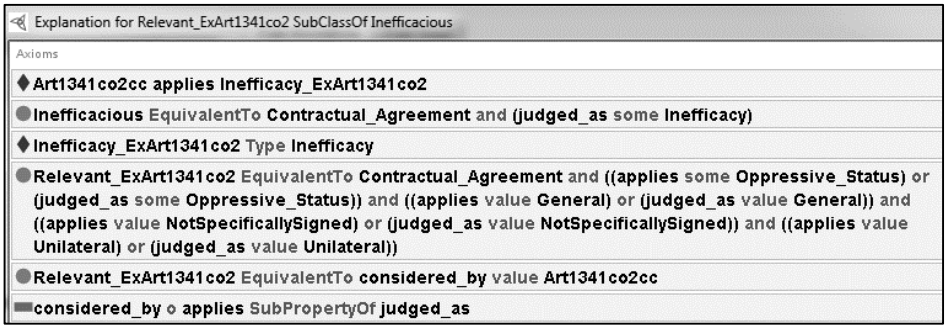

Fig. 27. Explanation for Relevancy being inferred as a subclass of Inefficacious.

Fig. 28. Stated and inferred property assertions on the exceptional contractual agreement.

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 which defeats the first one? As it should be clear, that issue has no straight solution inside DL, such as OWL-DL logics: introducing some negative condition for the rule to apply (in the form `if (not (exception))`), the open-world assumption OWL relies on would requires to explicitly state for each case that no exception applies. This would hinder the reasoning capabilities of the ontology library explained so far. A solution to this problem could rely on the modelling of the exceptional case as a subclass of the normal case, (see Figure 26). In this 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 allowing reasoning on exceptions without the need to rely on rules. The backside is that the classification of the circumstance as "exceptional" is added to the classification of inefficacy, not substituted to it (Figures 27 and 28). Again, this issue takes origin 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 stated in the ontology.

This issue represents the main reason why a complete syntactic modelling of legal rules is not feasible inside the ontology library, requiring instead a rule system (such as LKIF-Rules [23], Clojure, or LegalRuleML [40]) to be fully implemented. Nevertheless, the so-built ontology library represents the ideal background for such a rule system.

6. Conclusions

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 enhanced in legal reasoning through an argumentation theory. 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 characteristics of description logics. On the other hand, the ontology framework 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 [27], with added support for argumentation schemes.

References

[1] G. Antoniou, N. Dimaresis and G. Governatori, A Modal and Deontic Fegeasible 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, New York, 2009.

[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, 2009, pp. 11-20.
[6] E. Blomqvist, A. Gangemi and V. Presutti, Experiments in Pattern-based Ontology Design, in Proceedings of KCAP09, Los Angeles, ACM Press, 2009.
[7] F. Bobillo and U. Straccia, An OWL ontology for fuzzy OWL 2, in J. Rauch, Z.W. Ras, P. Berka and T. Elomaa, ads. Foundations of intelligent systems. Springer Berlin Heidelberg, 2009. 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, A Logical Understanding of Legal Interpretation, KR, 2010.
[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, Heidelberg, Springer, 2008, pp. 21-41.
[11] S. Brüninghaus and K. D. Ashley, Generating legal arguments and predictions from case texts, in ICAIL 2005, New York, 2005.
[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, 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 JURIX 2013, Springer, under publication.
[15] M. Ceci and T. F. Gordon, Browsing case law: An application of the Carneades Argumentation System, in Proceedings of the RuleML2012@ECAI Challenge, vol. 874, 2012, pp. 79-95.
[16] M. Ceci and M. Palmirani, "Ontology Framework for Judgement Modelling," in AI Approaches to the Complexity of Legal Systems. Models and Ethical Challenges for Legal Systems, Legal language and Legal Ontologies, Argumentation and Software Agents, LNCS vol. 7639, Berlin, Springer, 2012, pp. 116-130.
[17] S. Despres and S. Szulman, Construction of a Legal Ontology from a European Community Legislative text, in Legal Knowledge and Information Systems. Jurix 2004: The Seventeenth Annual Conference, Amsterdam, 2004.
[18] A. Gangemi, Ontology Design Patterns for Semantic Web Content, in International Semantic Web Conference 2005, pp. 262-276
[19] A. Gangemi, Design Patterns for Legal Ontology Construction, in: Trends in legal Knowledge. The Semantic Web and the Regulation of Electronic Social Systems, European Press Academic Publishing, 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
[21] A. Gangemi, Super-duper Schema: an OWL2+RIF DnS Pattern, in V. Chaudry, ed., *Proceedings of DeepKR Challenge Workshop* at KCAP11.
[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., Legal Ontologies and the Semantic Web, Springer, 2005.
[23] T. F. Gordon, Construting Legal Arguments with Rules in the Legal Knowledge Interchange Format, in Computable Models of the Law: Languages, dialogues, games, ontologies, Springer, Heidelberg, 2008, pp. 162-184.
[24] 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 (COMMA 06), Amsterdam, IOS Press, 2006.
[25] 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, 2009, pp. 137-146.
[26] T. F. Gordon, G. Governatori and A. Rotolo, Rules and Norms: Requirements for Rule Interchange Languages in the Legal Domain, in Rule Interchange and Applications, International Symposium, RuleML 2009, Berlin, Springer, 2009, pp. 282-296.
[27] G. Governatori and A. Rotolo, Defeasible Logic: Agency, Intention and Obligation, in Deontic Logic in Computer Science, Springer, 2004.
[28] M. Gruninger and M. Fox, The role of competency questions in enterprise engineering, in Proceedings of the IFIP WG5.7 Workshop on Benchmarking Theory and Practice, Trondheim, Norway, 1994
[29] R. Hoekstra, J. Breuker, M. Di Bello and A. Boer, LKIF Core: Principled Ontology Development for the Legal Domain, Law, Ontology and the Semantic Web, 2009, pp. 21-52.
[30] E. Kontopoulos, N. Bassiliades, G. Governatori and G. Antoniou, A Modal defeasible Reasoner of Deontic Logic for the Semantic Web, International Journal on Semantc Web adn Information Systems, 2011, pp. 18-43.
[31] W. Kusnierczyk, Nontological Engineering, in: International Conference on Formal Ontology in Information Systems (FOIS 2006), Springer, 2006.
[32] H. P. Lam and G. Governatori, The Making of SPINdle, in RuleML 2009, LNCS 5858, Berlin, Springer, 2009, pp. 315-322.
[33] D. T. Minh, T. Eiter and T. Krennwallner, Realizing Default Logic over Description Logic Knowledge Bases, in Proceeding of the ECSQARU, 2009.
[34] L. Mommers, Ontologies in the Legal Domain, in: Theory and Applications of Ontology: Philosophical Perspectives, Springer, 2010, pp. 265-276.
[35] D. Nute, Norms, Priorities, and Defeasible Logic, in Norms, Logics and Information Systems, IOS Press, 1998, pp. 201-218.
[36] M. Palmirani and F. Benigni, Norma-system: A legal information system for managing time, in Proceedings of the V Legislative XML Workshop, 2007, pp. 205-224.
[37] M. Palmirani and R. Brighi, Model Regularity of Legal Language in Active Modifications, LNCS, pp. 54-73, 2010.
[38] M. Palmirani, G. Contissa and R. Rubino, Fill the Gap in the legal Knowledge Modelling, in Proceedings of RuleML 2009, LNCS 5858, Berlin, Springer, 2009, pp. 305-314.
[39] M. Palmirani, M. Ceci, D. Radicioni and A. Mazzei, FrameNet model of the suspension of norms, in The 13th International Conference on Artificial Intelligence and Law, Proceedings of the Conference, Pittsburgh, 2011.

[40] M. Palmirani, G. Governatori, A. Rotolo, S. Tabet, H. Boley and A. Paschke, LegalRuleML: XML-Based Rules and Norms, in RuleML 2011, pp. 298-312.
[41] M. Palmirani, T. Ognibene and L. Cervone, Legal rules, text, and ontologies over time, in Proceedings of the RuleML@ ECAI 6th International Rule Challenge, Montpellier, 2012.
[42] M. Palmirani, L. Cervone, O. Bujor and M. Chiappetta, RAWE: An Editor for Rule Markup of Legal Texts, in RuleML 2013
[43] H. Prakken, Formalizing Ordinary legal Disputes: a Case Study, in AI&Law, 2008, pp. 333-359.
[44] Y. Ren, J. Z. Pan and Y. Zhao, Closed World Reasoning for OWL2 with NBox, Tsinghua Science & Technology, pp. 692-701, 2010.
[45] 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.
[46] G. Sartor, Legal Concepts as Inferential Nodes and Ontological Categories, Artificial Intelligence and Law, pp. 217-251, 2009.
[47] 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.
[48] J. R. Searle, Speech Acts: an Essay in the Philosophy of Language, Cambridge University Press, 1969.
[49] J. Shaheed, A. Yip and J. Cunningham, A Top-Level Language-Biased Legal Ontology, in LOAIT, Bologna, 2005.
[50] E. Sirin and B. Parsia, SPARQL-DL: SPARQL Query for OWL-DL, in 3rd OWL Experiences and Directions Workshop, 2007.
[51] F. Vitali, Akoma Ntoso Release Notes, 2011 [Online].
[52] W3C Consortium, "OWL 2 Web Ontology Language Overview," 11 December 2012. [Online]. Available: www.w3.org/TR/2012/REC-owl2-overview-20121211/.
[53] D. Walton, C. Reed and F. Macagno, Argumentation Schemes, Cambridge University Press, Cambridge, 2008.