AN OWL ONTOLOGY LIBRARY REPRESENTING JUDICIAL INTERPRETATIONS

Marcello Ceci[1]
Aldo Gangemi[2,3]

[1]CIRSFID, University of Bologna
[2]ISTC-CNR, Rome, Italy
[3]LIPN, Université Paris13-Sorbonne-Cité-CNRS, Paris, France

m.ceci@unibo.it
aldo.gangemi@cnr.it

Abstract: The paper introduces a formal model of judgments that, starting from the text of decisions, produces an OWL ontology that represents the interpretations performed by a judge while conducting a discourse towards an adjudication. The final goal of this method is to design an ontology framework capable of detecting and modelling jurisprudence directly from the text, and performing some basic reasoning on the resulting ontologies.

1. Introduction

Precedents 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 (or case-law) are a fundamental source for law interpretation, to the point that the exercise of jurisdiction can even influence the scope of the same norms it has to apply, both in common law and civil law legal systems – although to different extents. The AI & Law research community has gathered significant results on this topic since the 1980s, with different approaches: legal case-based reasoning [1][6], ontology-based systems [15], and more recently argumentation [9][10].

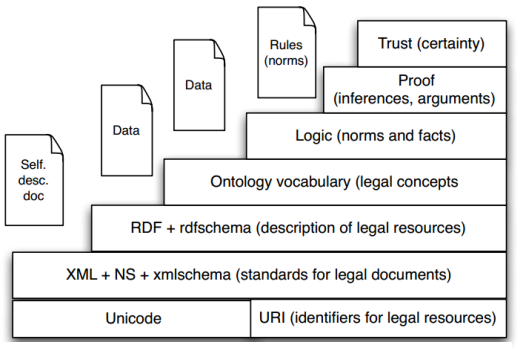

Fig. 1 - Tim Berners Lee's semantic web layer cake, adapted to the legal domain by G. Sartor [21]

The goal of the present research is to define a semantic web framework for precedent modelling, by using knowledge extracted from text, metadata, and rules [3][16]. Cornerstones of the framework are an ontology that represents the core structure of case-law, and the metadata connected with judicial legal concepts.

The ontology constitutes the basis of a semantic tool that enriches

the XML mark-up of precedents and supports legal reasoning. We believe that the new features of OWL2 could unlock useful reasoning for legal knowledge, especially if combined with rules.

The research relies on the previous efforts of the community in the field of legal knowledge representation [5] and rule interchange for applications in the legal domain [9]. The issue of implementing logics to represent judicial interpretation has already been faced [4], albeit only for the purposes of a sample case. The aim of the present research is to apply these theories to a set of real legal documents, stressing the OWL axioms definitions as much as possible in order to enable them to provide a semantically powerful representation of the legal document and a solid ground for an argumentation system using a defeasible subset of predicate logics.

Our task is to formalize the legal concepts and argumentation patterns contained in a judgment, in order to check, validate and reuse the discourse of a judge - and the argumentation he produces - as expressed by the text. In order to achieve this, we have used four different models:

a) a *document metadata structure*, capturing the main parts of the judgments, which create a bridge between text and semantic annotation of legal concepts;
b) a *legal core ontology*, modelling the abstract legal concepts and the institutions that capture the main parts of a rule of law [18];
c) a *legal domain ontology*, modelling the main legal concepts in a specific domain concerned by the case-law (e.g. contracts, e-commerce, tort law, etc.);
d) an *argumentation system* [7], modelling the structure of argumentation (arguments, counterarguments, premises, conclusions, rebuttal, etc.).

This article only introduces the structure of the core and domain ontologies - points b) and c) - which have been designed to organize the metadata coming from the text of judgment decisions, and to infer relevant knowledge about precedents. The approach is exemplified with reference to a sample of Italian case law. The metadata layer and argumentation systems - points a) and d) - are not described in the present work: the metadata layer relies on the Akoma Ntoso standard [2], while multiple solutions are being tested for building argumentation out of this ontology library: Carneades [7], SPINdle [14], and a Drools application currently under development [20].

Our approach for the ontology layer intends to satisfy the following functional requirements:

- *text-to-knowledge morphism*: we want to design the knowledge that can be extracted from a text (i.e. a 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 [17];
- *distinction between document layers*: we want the ontology to clearly distinguish between the medium and expression (the legal text) and its meaning (the legal concepts and rules contained in the text), so that more (and possibly inconsistent) legal meanings can correspond to a same legal text;
- *shallow reasoning on judgement's semantics*: we want the ontology to enable reasoning on the material circumstances, legal concepts and judicial interpretations

contained in precedents, deriving inferences out of the legal concepts and elements involved in a judicial decision;

- *querying*: being able to perform complex querying, e.g. by using SPARQL-DL [22], on qualified parts of a judgment text. For example, we want to make (successful) queries that encode a question such as: "*give me all the judgments 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*: we want to detect relevant parts of a judicial text by using semantic annotations based on judicial ontologies;
- *modularity*: the legal core ontology should define common concepts of all domain ontologies, which in turn should be automatically imported depending on the task;
- *supporting case-based reasoning*: we want to perform legal case-based reasoning [7] by using the ontology reasoner in combination with a set of rules and a rule engine.

Judicial ontologies are intended to create an environment where the knowledge extracted from the decision text can be processed and managed, and a deeper reasoning on the judicial interpretation grounding the decision itself is made possible. Deeper reasoning leads to the satisfaction of the following domain requirements:

- *finding relevant precedents* that are not explicitly cited in the decision;
- *validating the adjudications* of the judge on the claims brought forward by the parties during the trial on the basis of applicable rules, accepted evidence, and interpretation;
- *suggesting legal rules/precedents/circumstances* that could bring to a different adjudication of the claim.

The structure of the ontology library also aims at an efficacious scaling from legal concepts to factors, up to dimensions and legal principles: all these concepts can be represented in the domain ontology, and the hook of the judicial concepts to the core ontology should foster semantic alignment between differently designed domain ontologies (the current ontology library alignments have not yet been tested). Eventually, practical applications of the ontology library include:

- compliance check of contract drafts, i.e. through a plugin of a word processor using NLP techniques to recognize sentences and clauses that could be relevant under 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 (i.e. integrated into a word processor to help the judge while writing the judgement, 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;

- new 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, using a graphical interface and a formal argumentation structure instead of plain text.

2. Legal ontology design methods applied

Judicial ontologies are currently designed in two modules [18]:

- a *Core Ontology* describing the constituents of a precedent in terms of general concepts, through an extension to the LKIF-Core [5][13] legal ontology;
- a *Domain Ontology* representing the concepts and the rules expressed by the Italian "Codice del Consumo" (Consumer Code) and in artt. ("articles") 1241-1242 of the Italian Civil Code, as well as all relevant knowledge extracted from a set of Italian judgments 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 modelling the legal domain ontology, and top-down for modelling the core ontology classes and the argumentation theory components [15].

The approach adopted is based on a multi-layer paradigm, where the legal resource is managed in separated levels which are linked to each other but organized in order to allow multi-annotation, multi-interpretation, and multi-ontology with redundancy of representation. The syntactical approach was based on the following schema:

- Text annotation in XML: the Akoma Ntoso standard [2][23] grants proper mark-up of the structure of the judgement and of 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 judgments), but also the legal qualification of relevant parts of the decision, such as the minority report or the dissenting opinion;
- Ontology annotation: using external OWL definitions and linked through special mechanism to the XML document;
- 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 modelling, using the argumentation theory (see below).

Evaluation has been performed on a sample set of Italian case law of 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 involves situations where strictly deductive logic is not sufficient to represent the legal reasoning as performed by a judge. In particular, defeasible logics [12] 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 our research: in fact, OWL in general 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, we want to investigate how far OWL2 can be used in order to improve performance, computability, and management of classes in a defeasible logic context.

The software used to model the ontology (and from which the images of this paper are taken) is Protégé 4.1.0, supporting some of the features introduced by OWL2.

2.1 Judgement Structure

The judgment in Akoma Ntoso [23] is a particular type of document modelled for detecting the main significant parts of the precedent document: header for capturing main information such as parties, court, neutral citation, document identification number; body for representing the main part of the judgment, including the decision; conclusion for detecting the signatures.

The body is divided into four main blocks:

- the `introduction`, where usually (especially in common law decisions) the story of the trial is introduced;
- the `background`, dedicated to the description of the facts;
- the `motivation`, where the judge introduces the arguments supporting his decision;
- the `decision`, where the final outcome is given by the judge.

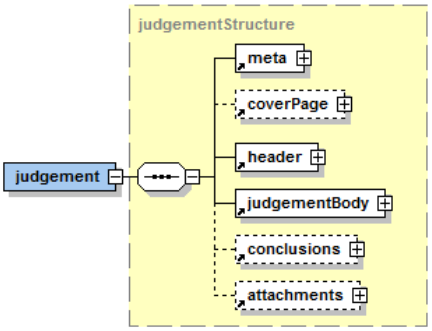

Fig. 2 – Judgment structure in Akoma Ntoso

This division is fundamental for detecting facts and factors from the background: in the `motivation` we detect arguments and counterarguments and in the `decision` the final conclusion of the legal argumentation process. Those qualified fragments of text should be annotated by legal experts with the help of a special editor tool (e.g. Norma-Editor) that allows an easily linking between text, metadata and ontology classes.

2.1 Core Ontology

The core ontology [1] introduces the main concepts and interactions in the legal domain, defining the classes which will be later filled with information taken from the judicial decisions. Even though the core ontology should be domain-generic and not modeled upon a specific legal subject, the model presented here was conceived to successfully represent the interaction in the civil law subject, when contracts, laws and judicial decisions come into play. Obviously, it will be necessary to add further classification prior to successfully expand the ontology library to a different domain (es. Public contracts, administrative law, tort law).

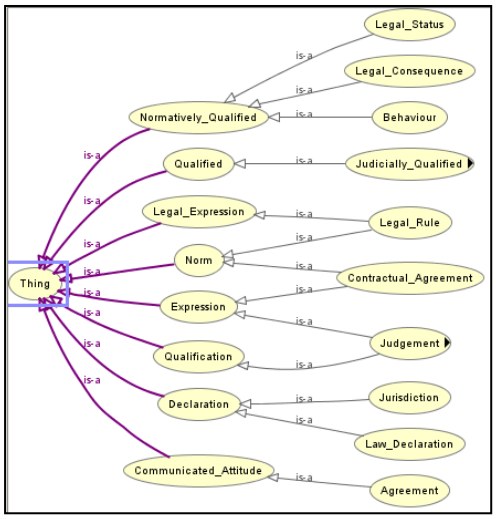

Fig. 3 – Core Ontology's specification of LKIF-Core

The backbone of the Core Ontology is represented by three LKIF-Core classes:

- `Qualificatory_Expression` (subclass of `Mental_Entity>Mental_Object>Proposition>Expression>Legal_Expression`) represents a legal expression which ascribes 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").
- `Qualification` (`Mental_Entity>Mental_Object>Proposition`) expresses e.g. a judgement: the thing qualified by the qualification is comparable to something else.
- `Qualified` represents anything which is the object of some qualification.

On the basis of those classes the following ontology design pattern [8] was built. Since the main object to be represented in the present set of ontologies is the normative/judicial qualification brought forward by performative utterances (contractual agreements, legal rules and judicial interpretations), the classes presented above constitute the nucleus of the Core Ontologies. The LKIF-Core `Qualification` and `Qualified` classes are linked only by a single property (qualifies/qualified_by), but what we rather want to model is an n-ary relation between (1) a qualifying expression, (2) the kind of qualification and (3) the object being qualified. In order to represent this, the property "qualifies" has been forked into two new properties: "considers" and "applies". The first one, "considers" (modeled as superclass of the LKIF-Core properties "evaluates", "allows", "disallows") represents the object of the qualification. The second property, "applies", shows *towards which concept* the qualification is made. For example, a

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

`Contractual_Agreement` considers a `Material_Circumstance` and applies a `Legal_Status`; a `Legal_Rule` considers a `Legal_Status` and applies a `Legal_Consequence`; a `Judicial_Interpretation` considers a `Material_Circumstance` and applies a `Legal_Status`; an `Adjudication` considers a `Judicial_Claim` and applies a `Judicial_Outcome`.

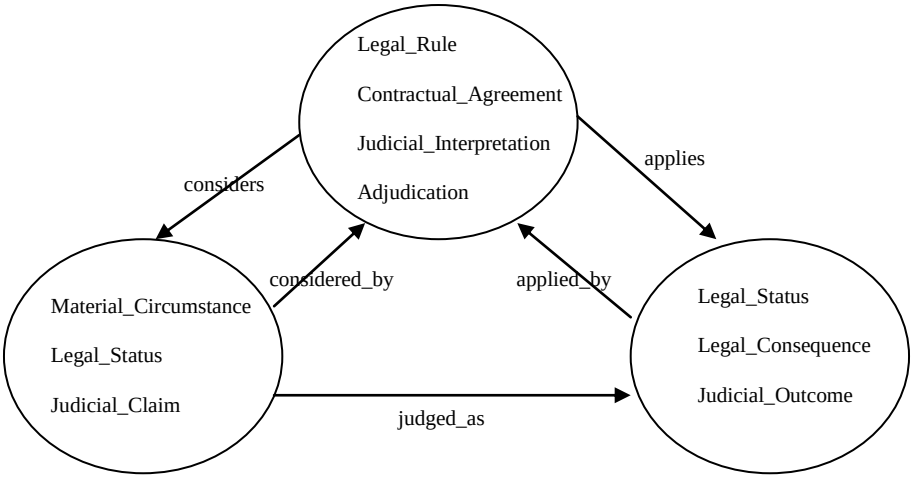

Fig. 4 – Interactions between qualifications, qualifiers, and qualified things.

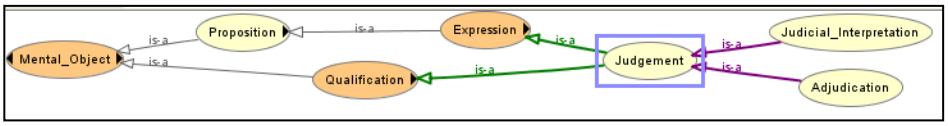

Fig. 5 – Visualization of the Judgement class as a Qualifying Legal Expression

Qualifying Legal Expressions - To overcome the limited expressivity of the original LKIF-Core classes a new ontology conceptual class called "Qualifying_Legal_Expression" has been conceived, putting together the characteristics of the `Qualificatory_Expression` and `Qualification` classes, enhanced by the fork of the qualifies property. This class represents the formalization of dispositions, such as the three legal expressions involved in contract law-related judicial decisions: `Contractual_Agreement`, `Legal_Rule` and

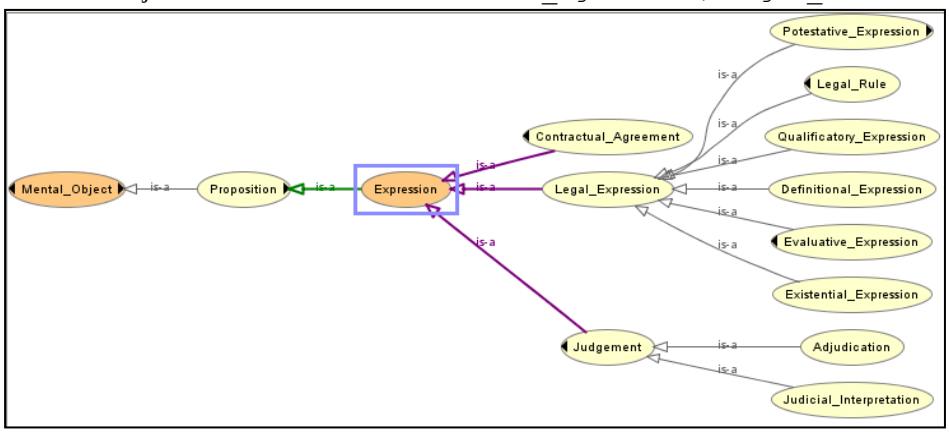

Fig. 6 – Visualization of the Expression class

`Judgement`.

As `Qualificatory_Expression` sub-classes, the Qualifying Legal Expressions contain all information related to their original "speech act": its semantic bonds with the externalization, the legal power and the agents ensure a complete representation of all aspects that may come into play when facing a legal issue (the legitimacy of the legislative body/court/legal party, the characteristics of the corresponding legal document, the identity/characteristics of people/bodies involved...). Their main properties are "medium" and "attitude" (see below for a specification of the `Medium`, `Attitude` and `Agent` classes).

As `Qualification` subclasses, the Qualifying Legal Expressions contain all information related to the effects they have in the legal world: the legal categories/obligations/legal effects they create, modify or repeal. A subdivision can be made between one direct subclass (`Judgement`, which in this perspective is furtherly divided into the `Judicial_Interpretation` and `Adjudication` subclasses) and two subclasses of `Norm` (`Legal_Rule` and `Contractual_Agreement`). As explained before, the property "qualifies" -

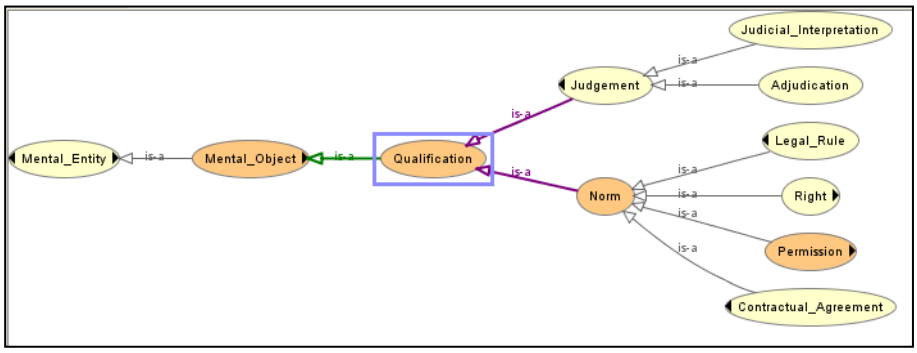

Fig. 7 – Visualization of the Qualification class

linking the qualifying expression to the `Qualified` expression - has been forked into two new properties: "considers" and "applies", representing respectively the direct object and the "destination" of the qualification.

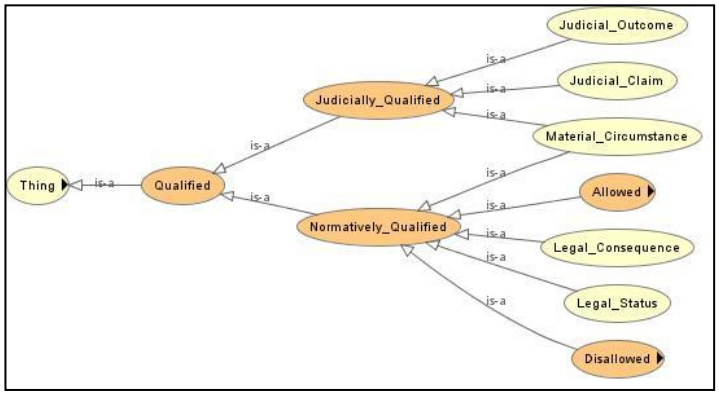

Fig. 8 - Visualization of the qualified class

Qualified Expressions - All the ranges of the "considers" and "applies" properties presented above are subclasses of the `Qualified` class. Its subclasses are `Normatively_Qualified`, a class already present in LKIF-Core, and `Judicially_Qualified`, created anew.

`Normatively_Qualified` expressions include `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 which 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`. As we will see, 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.

`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`.

The judged_as Property Chain - The miscellaneous elements that can be 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`. So, for example, a `Contractual_Agreement` can be considered_by some `Judicial_Interpretation` who applies some `Legal_Status` to it (i.e. the agreement is oppressive, is inefficacious, represents an arbitration clause, is specifically signed by both parties). In these cases, a OWL2 property chain directly links the `Contractual_Agreement` to the `Legal_Status` judicially applied to it. This "strong" link, represented by the property "judged_as", is the the fundamental information that we want to represent – and manage – through this set of ontologies.

Mediums, Propositional Attitudes and Agents - these LKIF-Core classes describe the background of an `Expression`. The `Medium` class identifies the support through which the proposition is expressed. It does not represent the material support

of the `Expression` instance but rather its *genus* (`Contract`, `Precedent`, `Code`). The `Propositional_Attitude` class was specified with the `Jurisdiction`, `Law_Declaration` and `Agreement` subclasses, representing the enabling powers that stand behind a `Judgement`, a `Legal_Rule` and a `Contractual_Agreement`, respectively. On the contrary, to represent the authors of a Qualifying Legal Expression there was no need to specify the subclasses of `Agent` already present in LKIF-Core (`Person` and `Organization`). This knowledge about agents and attitudes can be important in some judicial cases: i.e. if a claim is based on the lack of contractual power by one of the parties, or on the identity/characteristics of a part, or on the lack of force by some law or other regulation (which can in turn depend by the lack of legitimacy of one of its authors). Also the modeling of roles (already present in LKIF) can be very useful in representing critical factors of particular precedents.

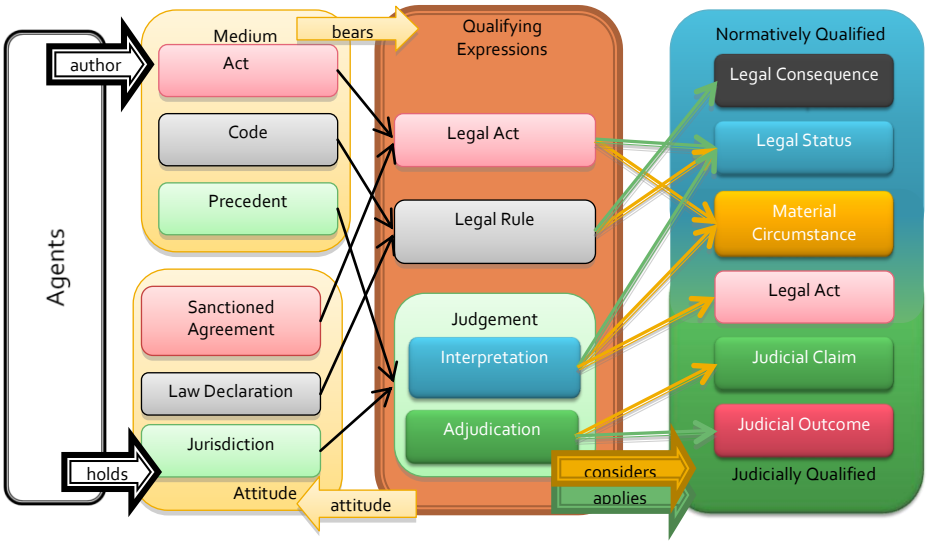

Fig. 9 – The Core Ontology graph.

Modularity of the Core Ontology - The expansion brought by the Core Ontology to the LKIF-Core concepts is currently oriented to the representation of the elements involved in civil-law cases regarding contract law. Nevertheless, the Core Ontology provides general – and relatively open - categories for this kind of judicial activity to be represented, and can therefore be considered as a core to be "expanded" with categorization from other branches of law, but not to be "substituted", since the basic concepts introduced here may come into play also in judgements concerning different subjects.

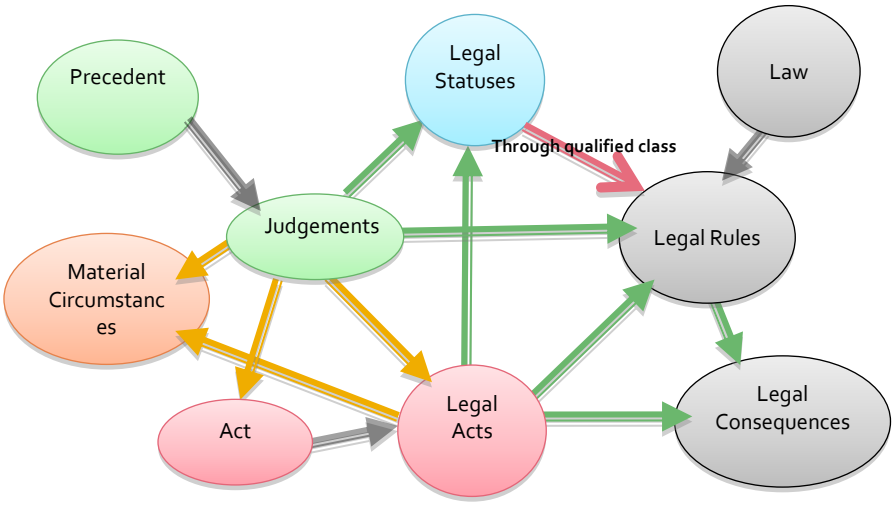

Fig. 10 – semantic relations between represented knowledge

2.2 Domain Ontology

Following this structure, the metadata taken from judicial documents are represented in the Domain Ontology[2]. 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 6.1). 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 modeling of the same concept by different authors).

Modeling of the law - 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 (fig. 11). The reasoner will infer knowledge about the rule, linking it (through the considers property) to the contractual agreements which fall under the scope of that norm.

[2] http://codexml.cirsfid.unibo.it/ontologies/judging_contracts_domain.owl

Fig. 11 - Stated property assertion of a Legal Rule instance

Fig. 12 – Visualization of the expression class, highlighting the subclasses of Contractual_Agreement introduced by the legal rules.

Legal rules are also represented through anonymous subclasses of the `Normatively_Qualified` class, called `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" (fig. 13). Please notice that, in the graph visualizer (fig. 12), 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

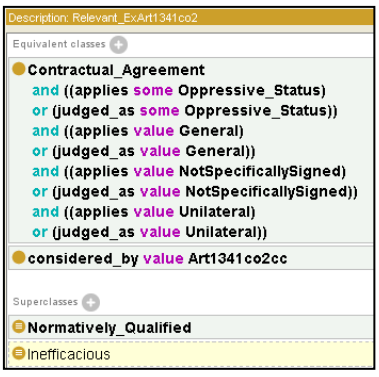

Fig. 13 - Axiom for classification of Contractual Agreements under Art. 1341co2

subdivisions which depend on the legal framework created by the legal rules of the domain.

Modeling of the contract - A contract is a composition of one or more `Contractual_Agreements` (a `Contract` for the whole, multiple `Contract_Clauses` for its parts), 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.

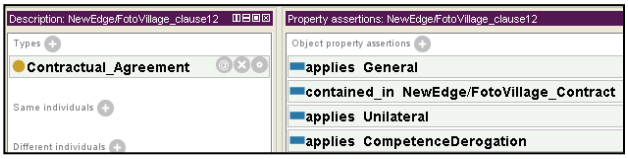

Fig. 14 - Description and property assertions of the contract clause's content.

In the actual model, the material circumstances considered by the contractual agreement were not included: that is, because this has no relevance when capturing the sheer interpretation instances these agreement undergo: it would rather become useful when delving deeper into the single interpretation, capturing the smaller factors which led to that specific interpretation.

Modeling of the decision -The `Judgement` class includes an instance identifying the case as a whole (the precedent) and several ones identifying its parts: at least an `Adjudication` and zero or more `Judicial_Interpretations`. They share a common attitude (a `Jurisdiction` power) a `Precedent` medium and some agents (claimant, defendant, and court). The `Adjudication` contains the `Judicial_Outcome` of the `Judicial_Claim`. (it considers the claim and applies the outcome), while the `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

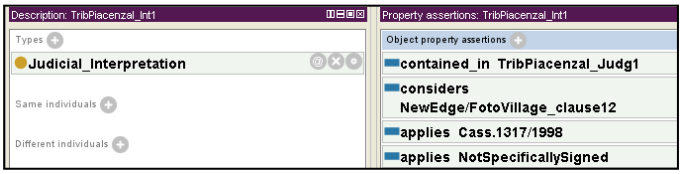

Fig. 15 - Description and property assertions of the judicial interpretation.

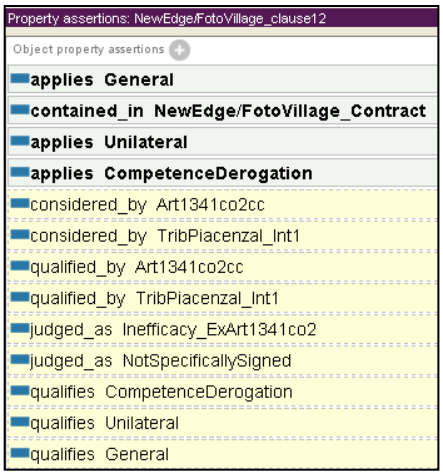

Fig. 16 - Inferred knowledge on the Contractual Agreement instance.

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 are.

Reasoning on the knowledge base - To check the consistency of this knowledge we will use Hermit 1.3.6[3] queries. 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 law rules 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 first objective for gathering all this semantically-rich information is advanced querying on precedents, but more can be achieved by combining different `Judicial_Interpretation`s with knowledge coming from the contract and the applicable law: the ontology reasoner is in fact capable of predicting – to some extents – the outcome of the judge (i.e. predicting that a clause will be judged as valid/invalid) and to run inferences about the agreement (i.e. as interpreted, the clause is irrelevant for the whole Italian Consumer Law/for the legal rule contained in article 1342 comma 2 of Italian Civil Code).

This inferred knowledge is important for two reasons: *a.* by "predicting" the judge's final statement on the clause (even if not that on the claim), this knowledge represents a logic and deontic check on the legal consequences the judge takes from its interpretation; *b.* it gives a fundamental element for the 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 judgment.

3. OWL2 Constructs Used

OWL 2 introduces several features to the original Web Ontology Language, some of which allow a richer representation of knowledge, mostly when dealing with properties and datatypes. Two features concerning properties have resulted useful in the design of the judicial ontologies:

Keys: An **HasKey** axiom states that each *named* instance of a class is uniquely identified by a (data or object) property or a set of properties - that is, if two named instances of the class coincide on values for each of key properties, then these two individuals are the same. This feature can be useful for identifying the unique "actors" of the judicial claim, such as the parties, the contract, the norm, and the decision itself.

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

Property Chains: The OWL 2 construct ObjectPropertyChain in a SubObjectPropertyOf axiom allows a property to be defined as the composition of several properties. Such axioms are known as *complex role inclusions* in SROIQ. In the present ontology library, the property chain "judged_as = considered_by o applies" is used in two different ways (in interpretations and rule applications) to create a strong interpretational link between a material circumstance and its 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 "weak" one introduced directly by the contract (represented by the property applies). Reasoners can therefore treat these two links accordingly. Secondarily, as already said, the legal rule axiom work through an "anonymous qualified class" which links all relevant expression 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 the qualified expression to the law, and from the law to the 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 exploitation of the OWL 2.0 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.

4. An Example of Precedent Modeling

The modeling of the ontology is explained here through a simple example of data insertion and knowledge management by the Domain Ontology:

In the decision given by the 1st section of the Court of Piacenza on July 9th, 2009[4], 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 who says "a general and unilateral clause conerning a 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.

4 Sent. N. 507 del 9 Luglio 2009, Tribunale di Piacenza, giudice dott. Morlini.

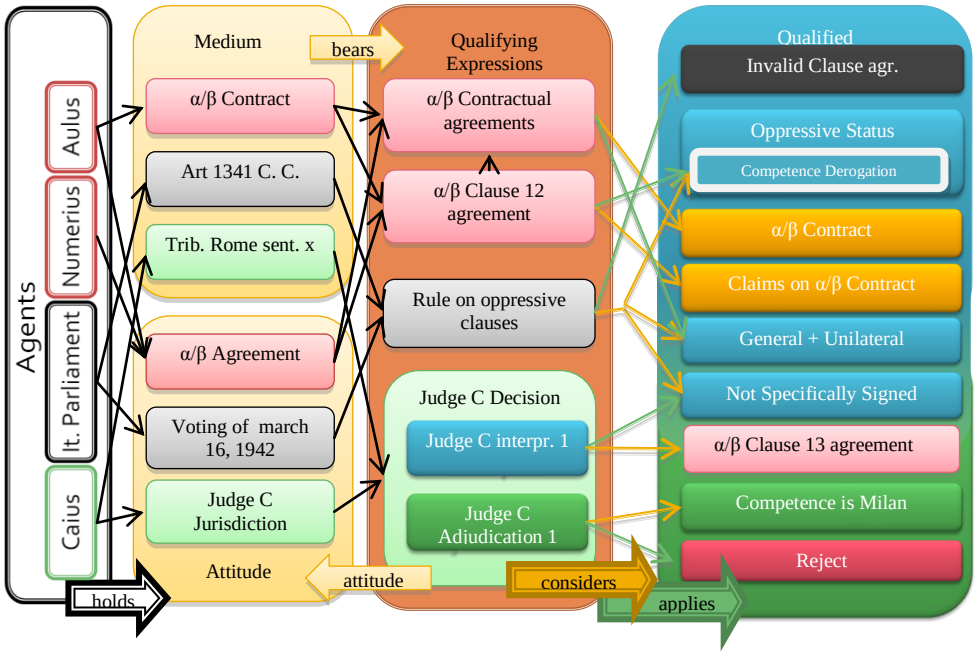

Fig. 17 - The example graph

In order to represent the knowledge contained in that judgment text, we need to model three documents: 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.

Modeling of the law – 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.*

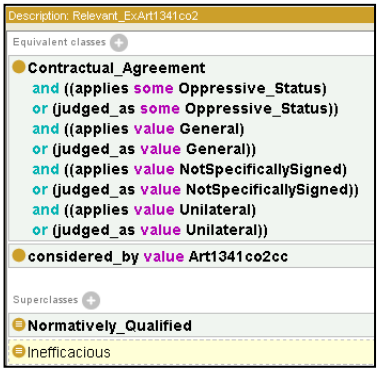

Fig. 18 - Description of the abstract class used to sort contractual agreements under the legal rule.

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`. 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, each `Contractual_Agreement` which applies "General", "Unilateral", "NotSpecificallySigned" and an `Oppressive_Status` will be considered_by

"art1341Co2", which in turn applies the Legal_Consequence of "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".

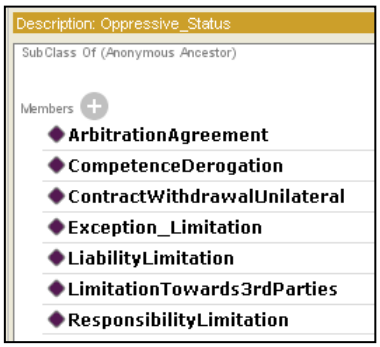

Fig. 19 - The list of legal statuses classified as oppressive.

Modeling of the contract clause- The Contract_Clause "α/βClause12" 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 the contractual parties have no power to force a legal status into a contract, and that reconducting 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.

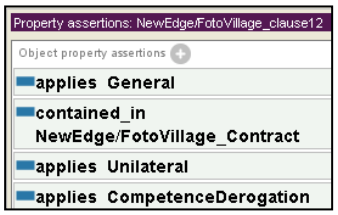

Fig. 20 - Stated property assertions for the sample agreement.

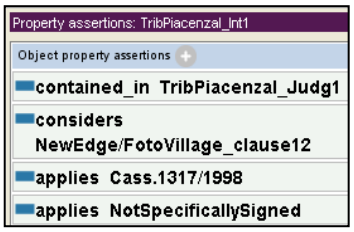

Fig. 21 - Stated property assertions of the sample judicial interpretation

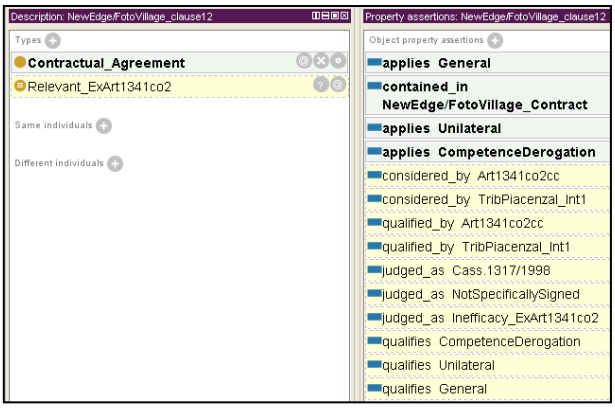

Fig. 22 - Inferred Description and property assertions of the contract clause's content.

Modeling of the decision - The Judgment instance is created, as well as its components (single interpretation instances, adjudication...). Among them, the "tribPiacenzaI_Int1" Judicial_Interpretation is created: it considers the Contractual_Agreement contained in "α/βClause12" and applies the

"notSpecificallySigned" `Legal_Status`. The instance contains also a reference to the precedent (Cass.1317/1998), which represent a semantically-searchable information on the interpretation instance.

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". 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, a simple property chain gives the clause its final (efficacy/inefficacy) status under that law.

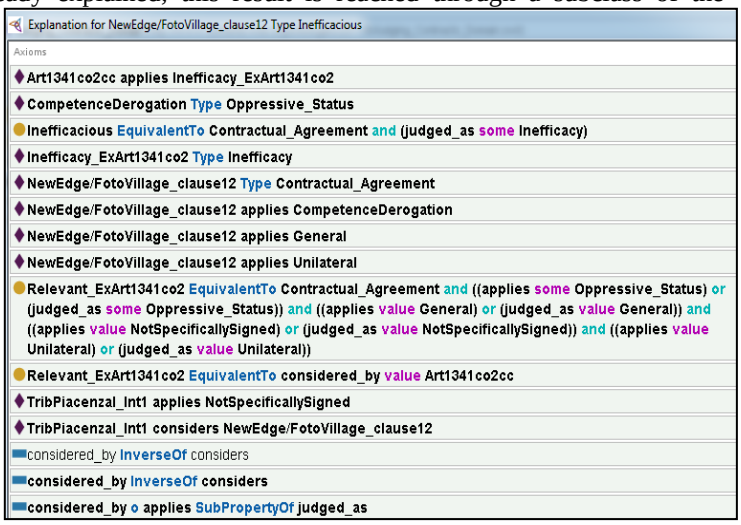

Fig. 23 - Explanation for the sample agreement being inefficacious.

5. Evaluation of the ontology library

The ontology library, in its sample taken from real judicial decisions, proved to meet the requirements of:

- *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. Moreover, the LKIF-Core's `Medium` class allows to represent different manifestations of the same expression;
- *shallow reasoning on judgement's semantics*: The property chain judged_as and the axioms for law relevancy 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 provides semantic sugar in this perspective;
- *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 case-based reasoning*: An argumentation system has been built on a "lite" version of the ontology library [7]. The axioms concerning law relevancy and law application have been 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.

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.

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). This could be improved by optimizing the reasoner and/or with a proper management of the ontology (and rules) modules.

6. Issues

6.1. The knowledge acquisition bottleneck

The modeling 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 should be annotated by legal experts: at the present time, this seems the only viable choice in the legal domain, as 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 [16]).

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) can allow 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.

6.2. Representing exceptions

A critical issue in representing the decision's content is represented by exceptions to legal rules. How can we 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: if we introduced some negative condition for the rule to apply (if (not (exception))), the open-world assumption OWL relies on would require us to explicitly state for each case that no exception applies. This would annihilate the reasoning capabilities of the ontology library we explained so far.

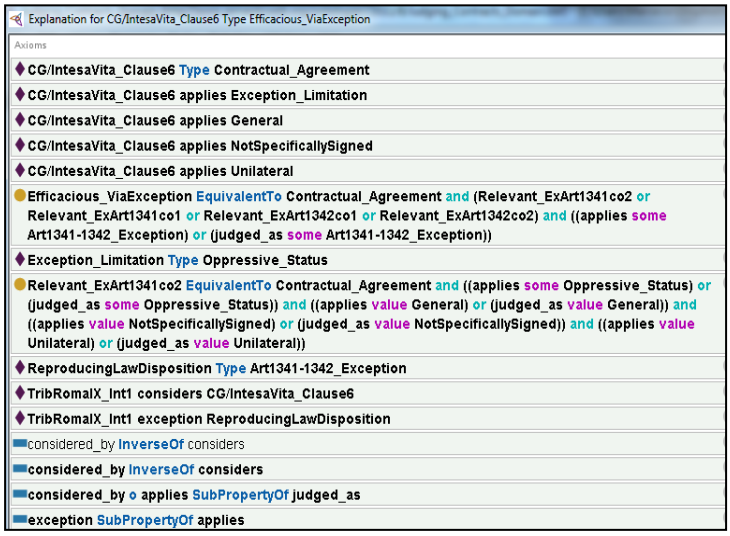

Fig. 24 - Explanation of a sample contract clause being not inefficacious

A solution to this problem could rely on the modeling of the exceptional case as a subclass of the normal case: this means that only the material circumstances which are relevant under the "regular" law can be classified as "exceptional".

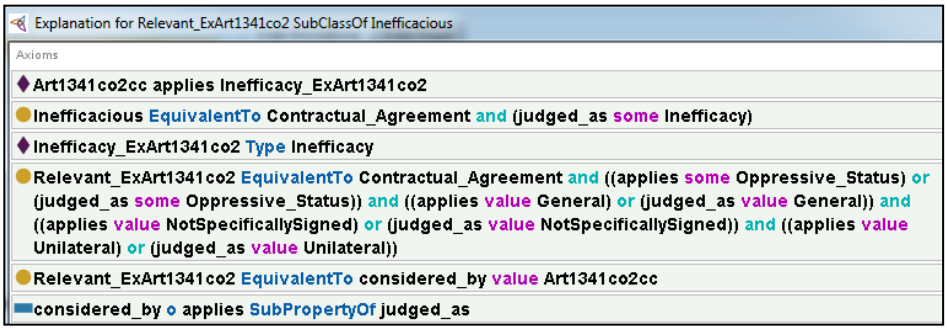

Fig. 25 - Explanation for Relevancy being inferred as a subclass of Inefficacious

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. Again, this issue takes origin from the open world assumption, and cannot be easily avoided while remaining inside OWL-DL: whenever we prevent the reasoner to "judge" a circumstance with a legal consequence, asking him to check that no exception exists,

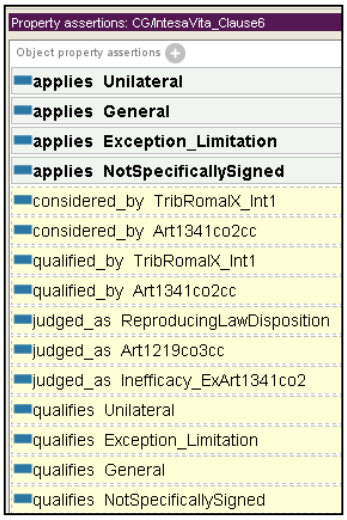

Fig. 26 - Stated and inferred property assertions on the "exceptional" contractual agreement.

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 reachable inside the ontology library, requiring instead a rule system (such as LKIF-Rules [11], Clojure, or LegalRuleML [19]) to be fully implemented. Nevertheless, the so-built ontology library represents the ideal background for such a rule system to work.

6. Conclusions

The ontology library presented in this paper 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 all the information available in the text to be enhanced in the legal reasoning through an argumentation theory. This approach allows to directly annotate the text with peculiar metadata representing the hook for the core, domain and argument ontologies. OWL2 is used to get as close as possible to the rules, in order to exploit the computational characteristic of description logics. On the other hand, the ontology framework has a strong weak point in the management of exceptions.

References

[1] Ashley K. D., *Ontological requirements for analogical, teleological, and hypothetical legal reasoning*. In: ICAIL 2009, pp. 1-10.

[2] Barabucci G., Cervone L., Palmirani M., Peroni S., Vitali F., *Multi-layer Markup and Ontological Structures in Akoma Ntoso*. In: LNCS 6237/2010, pp. 133-149.

[3] Bench-Capon T. J. M., Gordon T. F., *Isomorphism and argumentation*. In: ICAIL 2009, pp. 11-20.

[4] Boella G., Governatori G., Rotolo A., van der Torre L., *A Logical Understanding of Legal Interpretation*. In: KR 2010.

[5] Boer, A., Radboud, W., Vitali, F., *MetaLex XML and the Legal Knowledge Interchange Format*. In: Casanovas, P., Sartor, G., Casellas, N., Rubino, R. (eds.), *Computable Models of the Law*, Springer, Heidelberg (2008), pp. 21-41.

[6] Brüninghaus, S., Ashley K. D., *Generating legal arguments and predictions from case texts*. In: ICAIL 2005, New York, NY, USA, ACM Press, pp. 65–74.

[7] Ceci, M., Gordon, T.: *Browsing Case-Law: An Application of the Carneades Argumentation System*. In: Proceedings of the RuleML@ECAI 6th International Rule Challenge (2012).

[8] Gangemi, A., *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.

[9] Gordon, T. F., Governatori G., Rotolo A., *Rules and Norms: Requirements for Rule Interchange Languages in the Legal Domain*. In: RuleML 2009, pp. 282-296.

[10] Gordon, T. F., Walton, D.: *The Carneades Argumentation Framework: using presumptions and exceptions to model critical questions*. In: Dunne, P.E.: Computational Models

[11] Gordon, T. F., *Constructing Legal Arguments with Rules in the Legal Knowledge Interchange Format (LKIF)*. In: Computable Models of the Law, Languages, Dialogues, Games, Ontologies (2008), pp. 162-184.
[12] Governatori, G., Rotolo, A.: *Defeasible logic: Agency, intention and obligation*. In: Deontic logic in computer science, Springer (2004).
[13] Hoekstra R., Breuker J., Di Bello M., Boer A., The *LKIF Core Ontology of Basic Legal Concepts*. In: Casanovas P., Biasiotti M.A., Francesconi E., Sagri M.T. (eds), Proceedings of LOAIT 2007.
[14] Lam, H.P., Governatori, G.: *The making of SPINdle*. In: Governatori, G., Hall, J., Paschke, A. (eds.): RuleML 2009. LNCS 5858, 315–322.Springer, Berlin (2009).
[15] Mommers L., *Ontologies in the Legal Domain*. In: Poli R., Seibt J. (eds.), *Theory and Applications of Ontology: Philosophical Perspectives*, Springer 2010, pp. 265-276.
[16] Palmirani, M., Brighi, R., *Model Regularity of Legal Language in Active Modifications*. In: LNCS 6237/2010, pp. 54-73.
[17] Palmirani, M., Contissa, G., Rubino, R.: *Fill the Gap in the Legal Knowledge Modelling*. In: Proceedings of RuleML 2009, pp. 305-314.
[18] Palmirani, M., Ceci, M.: *Ontology framework for judgement modelling*. In: AICOL 2011 Proceedings. Springer, 2012 (under publication).
[19] Palmirani, M., Governatori, G., Rotolo, A., Tabet, S., Boley, H., Paschke, A.: *LegalRuleML: XML-Based Rules and Norms*. In: RuleML 2011, pp. 298-312
[20] Palmirani, M., Ognibene, T., Cervone, L.: *Legal Rules, Text and Ontologies over Time*. In: Proceedings of the RuleML@ECAI 6th International Rule Challenge (2012).
[21] Sartor G., *Legal Concepts as Inferential Nodes and Ontological Categories*. In Artif. Intell. Law 17(3) 2009, pp. 217-251.
[22] Sirin, E., Parsia, B.:, *SPARQL-DL: SPARQL Query for OWL-DL*. In 3rd OWL Experiences and Directions Workshop, 2007.
[23] Vitali F., *Akoma Ntoso Release Notes*. 1997. http://www.akomantoso.org.