



Deontic Reasoning for Legal Ontologies

Cheikh Kacfah Emani and Yannis Haralambous^(✉)

IMT Atlantique, Lab-STICC, UBL, 29238 Brest, France
{cheikh.kacfah,yannis.haralambous}@imt-atlantique.fr

Abstract. Many standards exist to formalize legal texts and rules. The same is true for legal ontologies. However, there is no proof theory to draw conclusions for these ontologically modeled rules. We address this gap by the proposal of a new modeling of deontic statements, and then we use this modeling to propose reasoning mechanisms to answer deontic questions i.e., questions like “Is it mandatory/permitted/prohibited to...”. We also show that using this modeling, it is possible to check the consistency of a deontic rule base. This work stands as a first important step towards a proof theory over a deontic rule base.

1 Introduction

Artificial intelligence for law is a prolific research area. In this area there have been many works on legal texts (e.g., legislation, regulations, contracts, and case law) modeling. This modeling pursues many goals, such as works on rule interchange issues, interoperability of rules' systems which lead to proposal of rules' format [1–4], legal ontologies construction [5,6], design principles [7,8], works on compliance checking problems [9,10] or automatic formalization of rules [11–13], etc. In legal texts which compose corpora for these works, we usually have deontic statements. Some rule representation standards include the modeling of deontic statements [2–4]. But there is no reasoning process using these deontic statements. For example using current standards there is no mechanism that one can use to answer questions like *Is fishing forbidden near the port of Brest?*, or to make inferences like *Since going left is forbidden then I must go straight*, etc. Our goal in this article is to propose mechanisms for answering deontic questions. Our proposal stands as a first attempt for a proof theory for ontologically modeled deontic rules. Our contributions are:

- An ontological modeling of deontic statement suitable for automatic reasoning on deontic rules.
- A first proposal of mechanisms for automatic answering of deontic questions towards a rule base.
- A first proposal of consistency checking process over a deontic rule base.

The rest of the paper starts with a presentation of related work in Sect. 2. Then, in Sect. 3, we introduce our work with a delimitation of the scope of rules we target. Next, in Sect. 4, we detail our modeling of rules. After that, we present

the mechanisms to answer a deontic question and we discuss consistency checking in Sect. 5 and we illustrate them with detailed examples in Sect. 6. Finally, we present future work in Sect. 7. A demo is available at <https://tinyurl.com/ydgwznba>.

2 Related Work

Rule modeling has been a very active field in the past years and still has a great interest for researchers. During the last years, many authors have proposed modeling languages and formalisms to represent rules and/or legal documents. We present some of their works and we conclude this section with a presentation of works on a proof theory for defeasible rules for propositional logic.

- Legal Knowledge Interchange Format (LKIF) is a specification that includes a legal core ontology and a legal rule language that can be used to deploy comprehensive legal knowledge management solutions. LKIF was developed in the ESTRELLA project [1, 14, 15]
- RuleML is a family of languages, whose modular system of schemas for XML favors web rule interchange [2, 16]. The family’s top-level distinction is deliberation rules and reaction rules. Deliberation rules are made of modal and derivation rules, which themselves include facts, queries, and Horn rules. Reaction rules include Complex Event Processing, Knowledge Representation, Event-Condition-Action rules, as well as Production rules. RuleML rules can combine all parts of both derivation and reaction rules. There are many engines built for rulebases for subsets of RuleML (OO jDREW¹, Prova², DR-Device³, NxBRE⁴, PSOATransRun⁵, etc.) *but none of them infers on deontic rule bases*.
- OWL Judge [17] allows to make deontic inferences on rules modelled in OWL DL by using the LKIF modelling. Using a standard OWL reasoner (Pellet in their case) authors can classify rule instances as “permitted” or “prohibited”. This is similar to our goal of deontic QA but we do not work only at instances’ level and we provide detailed pieces of information (exceptions, related cases, etc.) so that a user can really figure out what the legal texts state on its question.
- LegalRuleML is a standard (expressed with XML-schema and Relax NG) that is able to represent the particularities of the legal normative rules with a rich, articulated, and meaningful markup language [3]. LegalRuleML models defeasibility of rules and defeasible logic, deontic operators (e.g., obligations, permissions, prohibitions, rights), temporal management of rules and temporality in rules, classification of norms (i.e., constitutive, prescriptive), jurisdiction

¹ <http://www.jdrew.org/oojdrew/>.

² <https://www.prova.ws/>.

³ <http://lpis.csd.auth.gr/systems/dr-device.html>.

⁴ <https://nxbre.soft112.com/>.

⁵ http://wiki.ruleml.org/index.php/PSOA_RuleML#Implementation.

of norms, isomorphisms between rules and natural language normative provisions, identification of parts of the norms (e.g., bearer, conditions), authorial tracking of rules, etc. Concerning the rules themselves, LegalRuleML models the two main types of rules: (i) *constitutive rules* which define concepts or constitute activities that cannot exist without such rules, and (ii) *prescriptive rules* which regulate actions by making them obligatory, permitted, prohibited, recommended. The general shape of a LegalRuleML rule is `body \implies head` where, for both constitutive and prescriptive rules, where `body` is a set of formulas and deontic formulas and `head` is a formula (more precisely, the `head` of a constitutive rule is a formula and the one of a prescriptive rule is a set of deontic formulas). In this work, we take advantage of the rich modeling of normative documents within the LegalRuleML ontology and we propose an alternative to the ontological modeling of prescriptive rules. Using this new modeling, we propose mechanisms to answer deontic questions; they stand as a first step for the proposal of a proof theory for ontologically modeled deontic rules.

- To answer deontic questions, we take into account the defeasibility of rules (i.e., rules can be defeated by stronger ones) and the existence of superiority relation between conflicting rules. Antoniou *et al.* [18] present a proof theory for defeasible logic, in the frame for propositional logic. They claim that a conclusion in a defeasible theory D is a tagged literal that can have one of the following forms: $+Δq$ if the proposition q is *definitely provable* in D , $+δq$ if q is *defeasibly provable* in D , $-Δq$ if it is proved that q is *not definitely provable* in D and $-δq$ if it is proved that q is *not defeasibly provable* in D . We will see that for a deontic question, we can answer definitely yes/no with or without exceptions or we can answer yes/no for specific cases only, with or without exceptions; there are also some cases where the answer is “unknown,” and finally cases for which the reasoner identifies inconsistencies.

3 Scope of the Modeling

As we mentioned in the related work section, our modeling is in line with LegalRuleML ontology (LRO). Our focus is on rule modeling for deontic reasoning. Our concern is the rules themselves and not the ecosystem around rules i.e., temporal data, jurisdiction, authorial tracking of rules, etc., since they are modeled in ontologies like LRO. The questions we intend to answer are:

1. How can we model rules to ease automated legal reasoning on these rules?
2. How can we perform automated reasoning on deontic rules?

In LegalRuleML specifications, the body of rules can have both simple formulas and deontic formulas. Also, the head of rules can be either a simple formula for constitutive rules or a set of deontic formulas for prescriptive rules. Here we consider *prescriptive rules with no deontic formulas in the body part*. A generalization to other types of rules is part of our future work (see Sect. 7).

4 Rule Modeling

We model a rule as a four-dimensional object: We consider that a rule is made of *target(s)*, *condition(s)*, *context(s)* and *requirement(s)*. We provide the definition of these terms and illustrate them using the rule *The captain of any ship, called upon to assist or tow a ship carrying oil or gaseous hydrocarbon residues in French territorial waters, shall immediately inform CROSS*.⁶

- **Target.** The target of a rule is the entity towards which the rule is directed. It is the answer to the question “Who is the rule aimed at?”. In our example, the target is *The captain of any ship*.
- **Requirement.** The requirement is what the target should do. In the example, the requirement is *shall immediately inform CROSS*.
- **Context.** To find the context of a rule we answer the question “In which situation should be the target to execute the requirement of the rule?”. In our example, the context is that *the captain of the ship being called upon to assist or tow a ship carrying oil or gaseous hydrocarbon residues*.
- **Condition.** Conditions are criteria that must be fulfilled so that the target is able to execute the requirement. In our example, the condition is that *the ship (calling for assistance) facing an emergency is in French territorial waters*.

We use the class `shom:DeonticRule`, the properties `shom:ruleTarget`, `shom:ruleContext`, `shom:ruleRequirement`, `shom:ruleCondition` and `shom:deonticAction` to model a rule. We use these entities to extend LRO as we show in Fig. 1⁷.

In addition to the description of rules, LRO is able to describe the superiority relation that can exist between rules. Indeed, in some legal corpora we find conflicting rules and the ability of having a superiority relation avoids inconsistencies. In Sect. 6 we provide detail examples where we can see how LRO models the superiority relation and how this relation impacts inferences. It is crucial to have such priority relation to drive the right conclusion from a rule base.

5 Answering Deontic Questions

In this section, our goal is to draw a conclusion to answer the generic question “Is it OP to follow the requirement \mathcal{R} for target \mathcal{T} within the context \mathcal{C} and under the condition \mathcal{K} ? ” where OP denotes the correct form of one of the deontic terms *mandatory*, *permitted*, *prohibited* or *recommended*. We introduce this section with notation elements and mechanisms to answer a deontic question (Sect. 5.1) and then we detail the concepts introduced to present these mechanisms (Sects. 5.2 and 5.3).

⁶ Excerpt from Article 10 of arrêté <https://tinyurl.com/arrete2002>. CROSS is a Regional Operational Centre for Monitoring and Rescue.

⁷ The prefixes are the following: `owl:<http://www.w3.org/2002/07/owl>`, `<http://example.com/>`, `rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns>`, `rdfs:<http://www.w3.org/2000/01/rdf-schema>`, `shom:<http://example.com/shom>`, `lrmlmm:<http://docs.oasis-open.org/legalruleml/ns/v1.0/metamodel>`,

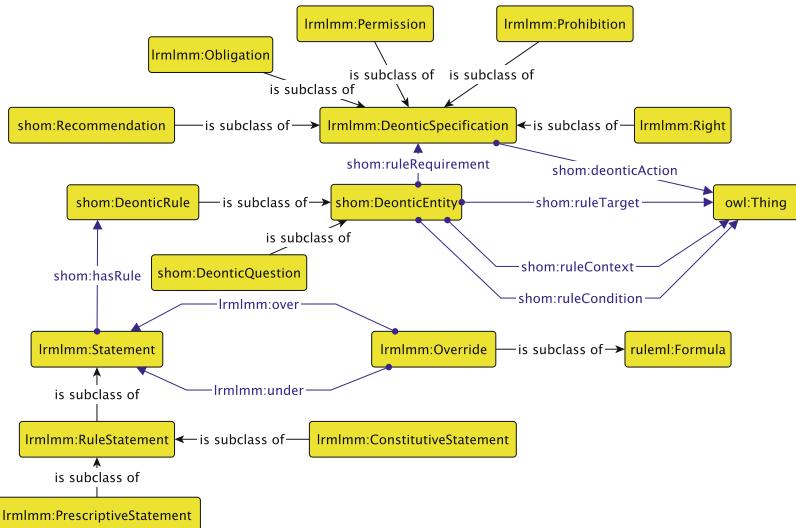


Fig. 1. The Graffoo diagram summarizing the extension of LegalRuleml ontology.

5.1 Drawing a Deontic Conclusion

We use the following notation:

- $\mathcal{Q}_{C,K,T,\text{OP},\mathcal{R}}$ represents the *question*: “Is it OP to follow the requirement \mathcal{R} for target T within the context C and under conditions K ”. We use \mathcal{Q} for the shorten form of $\mathcal{Q}_{C,K,T,\text{OP},\mathcal{R}}$
- $+c_{C,K,T,\text{OP},\mathcal{R}}$ denotes the fact that we conclude “yes” to $\mathcal{Q}_{C,K,T,\text{OP},\mathcal{R}}$. $-c_{C,K,T,\text{OP},\mathcal{R}}$ stands for the answer “no” to $\mathcal{Q}_{C,K,T,\text{OP},\mathcal{R}}$. Their shortened forms are $+c$ and $-c$. We write $+c$ or $-c$ when the “yes” or “no” answer *partially holds*, i.e., only for specific cases (to be determined in practice). $+c_E$, $-c_E$, $+c_E$ and $-c_E$ denote the fact that $+c$, $-c$, $+c$ and $-c$ are augmented with the list of exceptions.
- $\Delta\text{Pros}(\mathcal{Q})$ is the set of rules that *support a “yes”* to \mathcal{Q} , i.e., that support a conclusion $+c$. $\Delta\text{Cons}(\mathcal{Q})$ is the set of rules that *support a “no”* to \mathcal{Q} , i.e., that elements of $\Delta\text{Cons}(\mathcal{Q})$ support $-c$.
- For two sets of rules R_1 and R_2 , $R_1 > R_2 \Leftrightarrow \forall r_2 \in R_2, \exists r_1 \in R_1 : r_1 \neq r_2$ and $r_1 > r_2$ (i.e. r_1 overrules r_2).
- $\partial\text{Pros}(\mathcal{Q})$ is the set of rules that *partially support a “yes”* to \mathcal{Q} , i.e., that support a conclusion $+c$. Similarly, $\partial\text{Cons}(\mathcal{Q})$ is the set of rules that *partially support a “no”* to \mathcal{Q} , i.e., that support a conclusion $-c$.

To answer the question $\mathcal{Q}_{C,K,T,\text{OP},\mathcal{R}}$ (a.k.a. \mathcal{Q}), we proceed as follows:

1. We build $\Delta\text{Pros}(\mathcal{Q})$ and $\Delta\text{Cons}(\mathcal{Q})$. Hence, four scenarios can arise:
 - (a) $\Delta\text{Pros}(\mathcal{Q}) \neq \emptyset$ and $\Delta\text{Cons}(\mathcal{Q}) \neq \emptyset$.
 - If $\Delta\text{Pros}(\mathcal{Q}) > \Delta\text{Cons}(\mathcal{Q})$ then $+c_E$

- Else if $\Delta\text{Cons}(\mathcal{Q}) > \Delta\text{Pros}(\mathcal{Q})$, then $-\mathfrak{C}_E$
- Else "unknown (potential inconsistencies)"

For $+\mathfrak{C}_E$ and $-\mathfrak{C}_E$, the procedure to get exceptions is given in part Sect. 5.3

- (b) $\Delta\text{Pros}(\mathcal{Q}) \neq \emptyset$ and $\Delta\text{Cons}(\mathcal{Q}) = \emptyset$. We conclude $+ \mathfrak{C}_E$
 - (c) $\Delta\text{Pros}(\mathcal{Q}) = \emptyset$ and $\Delta\text{Cons}(\mathcal{Q}) \neq \emptyset$. We conclude $- \mathfrak{C}_E$
 - (d) $\Delta\text{Pros}(\mathcal{Q}) = \emptyset$ and $\Delta\text{Cons}(\mathcal{Q}) = \emptyset$. We go to step (2)
2. If $\Delta\text{Pros}(\mathcal{Q}) = \Delta\text{Cons}(\mathcal{Q}) = \emptyset$, we find if there are rules which support the partial conclusions $+ \mathfrak{c}$ or $- \mathfrak{c}$. So, we build $\partial\text{Pros}(\mathcal{Q})$ and $\partial\text{Cons}(\mathcal{Q})$ and:
 - (a) If $\partial\text{Pros}(\mathcal{Q}) \neq \emptyset$ and $\partial\text{Cons}(\mathcal{Q}) \neq \emptyset$: we conclude both $+ \mathfrak{c}_E$ and $- \mathfrak{c}_E$
 - (b) If $\partial\text{Pros}(\mathcal{Q}) \neq \emptyset$ and $\partial\text{Cons}(\mathcal{Q}) = \emptyset$. We conclude $+ \mathfrak{c}$ (it cannot have any exception, since at this stage $\partial\text{Cons}(\mathcal{Q}) = \Delta\text{Cons}(\mathcal{Q}) = \emptyset$)
 - (c) $\partial\text{Pros}(\mathcal{Q}) = \emptyset$ and $\partial\text{Cons}(\mathcal{Q}) \neq \emptyset$. We conclude $- \mathfrak{c}$
 - (d) $\partial\text{Pros}(\mathcal{Q}) = \emptyset$ and $\partial\text{Cons}(\mathcal{Q}) = \emptyset$. Since $\Delta\text{Pros}(\mathcal{Q}) = \Delta\text{Cons}(\mathcal{Q}) = \emptyset$ we answer "unknown"

In the next sections, we detail the construction of sets $\Delta\text{Pros}(\mathcal{Q})$, $\Delta\text{Cons}(\mathcal{Q})$, $\partial\text{Pros}(\mathcal{Q})$ and $\partial\text{Cons}(\mathcal{Q})$ (Sect. 5.2) and the way we find exceptions to a conclusion (Sect. 5.3).

5.2 Building of $\Delta\text{Pros}(\mathcal{Q})$, $\Delta\text{Cons}(\mathcal{Q})$, $\partial\text{Pros}(\mathcal{Q})$ and $\partial\text{Cons}(\mathcal{Q})$

In this section we give the formal expressions of $\Delta\text{Pros}(\mathcal{Q})$, $\Delta\text{Cons}(\mathcal{Q})$, $\partial\text{Pros}(\mathcal{Q})$ and $\partial\text{Cons}(\mathcal{Q})$. These expressions use the following notation:

- $A \mathfrak{a} B$ represents the assertion " $A \text{ rdf:type } B$ ".
- $A \subseteq^+ B$ means one of " $A \text{ rdfs:subClassOf } B$," " $A \text{ owl:sameAs } B$," " $A \mathfrak{a} B$ " is true. In this case we say that A is included in B or A is narrower than B or B contains A or B is broader than A . The relation \subseteq^+ is reflexive
- $A \subseteq^\tau B$ means " $\exists C$ so that $C \subseteq^+ A$ and $C \subseteq^+ B$ " is true. A specific case of this relation is when A and B are both superclasses of a third class C . In this case, we say that A and B share a common sub-entity.
- $A \subseteq^* B$ means one of " $A \subseteq^+ B$ " or " $B \subseteq^+ A$ " or " $A \subseteq^\tau B$ " is true. In this case we say that A is related to B .
- $(A_1, \dots, A_i, \dots, A_n) \mathfrak{R} (B_1, \dots, B_i, \dots, B_n) \equiv \forall i \in [1, n], A_i \mathfrak{R} B_i$, where \mathfrak{R} represents the relations \subseteq^+ or \subseteq^* or \subseteq^τ .
- We call *parameters of an entity* the quadruple made of its context \mathcal{C} , its condition \mathcal{K} , its target \mathcal{T} and its requirement \mathcal{R} . Here the term "entity" refers to a question $\mathcal{Q}_{\mathcal{C}, \mathcal{K}, \mathcal{T}, \mathcal{OP}, \mathcal{R}}$, or to a conclusion $\pm \mathfrak{C}_{\mathcal{C}, \mathcal{K}, \mathcal{T}, \mathcal{OP}, \mathcal{R}}$ or to an instance r of `shom:DeonticRule`. We note the parameters of an entity E as $(\mathcal{C}_E, \mathcal{K}_E, \mathcal{T}_E, \mathcal{R}_E)$.
- We define the notion of *conflicting deontic operator*. For each type of deontic operator, Table 1 gives the ones that are in conflict with it.

To understand Table 1 let us take for instance the first deontic operator: $\neg(\text{PER}) = \{\text{PRO}\}$ means that everything that is prohibited (in a given context, conditions, and for a given target), is not permitted (in that same context, under those same conditions and for that same target).

Table 1. Operations on deontic operators. PER: permission, PRO: prohibition, OBL: obligation, REC: recommendation, RIG: right

Deontic operator (OP)	Conflicting deontic operator ($\neg(\text{OP})$)
PER	{PRO}
PRO	{PER, REC, OBL, RIG}
OBL	{PRO}
REC	{PRO}
RIG	{PRO}

Definition 1. The **sup-rules** of $(\mathcal{C}, \mathcal{K}, \mathcal{T}, \text{OP}, \mathcal{R})$, are the rules having a deontic operator equal to OP and whose parameters are broader than $(\mathcal{C}, \mathcal{K}, \mathcal{T}, \mathcal{R})$, i.e.,

$$\text{sup-rules}(\mathcal{C}, \mathcal{K}, \mathcal{T}, \text{OP}, \mathcal{R}) = \{r, (\mathcal{C}, \mathcal{K}, \mathcal{T}, \mathcal{R}) \subseteq^+ (\mathcal{C}_r, \mathcal{K}_r, \mathcal{T}_r, \mathcal{R}_r) \wedge \text{OP}_r = \text{OP}\}.$$

Definition 2. The **neg-sup-rules** of $(\mathcal{C}, \mathcal{K}, \mathcal{T}, \text{OP}, \mathcal{R})$, are the rules having a deontic operator that conflicts OP and whose parameters are broader than $(\mathcal{C}, \mathcal{K}, \mathcal{T}, \mathcal{R})$, i.e.,

$$\begin{aligned} \text{neg-sup-rules}(\mathcal{C}, \mathcal{K}, \mathcal{T}, \text{OP}, \mathcal{R}) &= \{r, (\mathcal{C}, \mathcal{K}, \mathcal{T}, \mathcal{R}) \subseteq^+ (\mathcal{C}_r, \mathcal{K}_r, \mathcal{T}_r, \mathcal{R}_r) \wedge \text{OP}_r \in \neg(\text{OP})\} \\ &= \bigcup_{\text{OP}' \in \neg(\text{OP})} \text{sup-rules}(\mathcal{C}, \mathcal{K}, \mathcal{T}, \text{OP}', \mathcal{R}). \end{aligned}$$

The **sup-rules** of $(\mathcal{C}, \mathcal{K}, \mathcal{T}, \text{OP}, \mathcal{R})$ are the rules that (fully) endorse the veracity of the assertion *In context \mathcal{C} , and under conditions \mathcal{K} , the target \mathcal{T} is OP to (do/have/follow) \mathcal{R}* . This endorsement relies on the fact that each parameter of a rule of $\text{sup-rules}(\mathcal{C}, \mathcal{K}, \mathcal{T}, \text{OP}, \mathcal{R})$ is broader than its corresponding parameter in $(\mathcal{C}, \mathcal{K}, \mathcal{T}, \mathcal{R})$ and that the operator of the rule equals OP . On the other hand, **neg-sup-rules** are the rules that counter-attack the aforementioned assertion.

Definition 3. The **rel-rules** of $(\mathcal{C}, \mathcal{K}, \mathcal{T}, \text{OP}, \mathcal{R})$ are the rules having a deontic operator equal to OP and whose parameters are related to $(\mathcal{C}, \mathcal{K}, \mathcal{T}, \mathcal{R})$, i.e.,

$$\text{rel-rules}(\mathcal{C}, \mathcal{K}, \mathcal{T}, \text{OP}, \mathcal{R}) = \{r, (\mathcal{C}, \mathcal{K}, \mathcal{T}, \mathcal{R}) \subseteq^* (\mathcal{C}_r, \mathcal{K}_r, \mathcal{T}_r, \mathcal{R}_r) \wedge \text{OP}_r = \text{OP}\}.$$

Definition 4. The **neg-rel-rules** of $(\mathcal{C}, \mathcal{K}, \mathcal{T}, \text{OP}, \mathcal{R})$ are the rules having a deontic operator that conflicts OP and whose parameters are related to $(\mathcal{C}, \mathcal{K}, \mathcal{T}, \mathcal{R})$, i.e.,

$$\begin{aligned} \text{neg-rel-rules}(\mathcal{C}, \mathcal{K}, \mathcal{T}, \text{OP}, \mathcal{R}) &= \{r, (\mathcal{C}, \mathcal{K}, \mathcal{T}, \mathcal{R}) \subseteq^* (\mathcal{C}_r, \mathcal{K}_r, \mathcal{T}_r, \mathcal{R}_r) \wedge \text{OP}_r \in \neg(\text{OP})\} \\ &= \bigcup_{\text{OP}' \in \neg(\text{OP})} \text{rel-rules}(\mathcal{C}, \mathcal{K}, \mathcal{T}, \text{OP}', \mathcal{R}). \end{aligned}$$

A rule ρ is an element of $rel\text{-}rules(\mathcal{C}, \mathcal{K}, \mathcal{T}, \text{OP}, \mathcal{R})$ if each of its parameters $(\mathcal{C}_\rho, \mathcal{K}_\rho, \mathcal{T}_\rho, \mathcal{R}_\rho)$ is related, i.e., is broader or narrower than, or shares a common sub-entity with, the corresponding parameter from $(\mathcal{C}, \mathcal{K}, \mathcal{T}, \mathcal{R})$. So when we put all parameters of ρ together, either: (i) $(\mathcal{C}, \mathcal{K}, \mathcal{T}, \mathcal{R}) \subseteq^+ (\mathcal{C}_\rho, \mathcal{K}_\rho, \mathcal{T}_\rho, \mathcal{R}_\rho)$ or (ii) there is at least a parameter of ρ that is *strictly narrower than*⁸ the respective one in $(\mathcal{C}, \mathcal{K}, \mathcal{T}, \mathcal{R})$. Hence, “related” rules in case (i) are also member of $sup\text{-}rules(\mathcal{C}, \mathcal{K}, \mathcal{T}, \text{OP}, \mathcal{R})$ and those in case (ii) support partially, because of their “strictly narrow” parameters, the assertion behind $(\mathcal{C}, \mathcal{K}, \mathcal{T}, \text{OP}, \mathcal{R})$: we use them to define ∂Pros .

A similar discourse holds for $neg\text{-}rel\text{-}rules(\mathcal{C}, \mathcal{K}, \mathcal{T}, \text{OP}, \mathcal{R})$.

Definition 5. We say that a rule r (**fully**) supports a “yes” to \mathcal{Q} if (i) $\text{OP}_r = \text{OP}_{\mathcal{Q}}$ and (ii) $(\mathcal{C}_{\mathcal{Q}}, \mathcal{K}_{\mathcal{Q}}, \mathcal{T}_{\mathcal{Q}}, \mathcal{R}_{\mathcal{Q}}) \subseteq^+ (\mathcal{C}_r, \mathcal{K}_r, \mathcal{T}_r, \mathcal{R}_r)$. In other words, if r belongs to $sup\text{-}rules(\mathcal{Q})$. Hence,

$$\Delta\text{Pros}(\mathcal{Q}) = sup\text{-}rules(\mathcal{Q}). \quad (1)$$

Definition 6. We say that a rule r (**fully**) supports a “no” to \mathcal{Q} if (i) $r \in neg(\mathcal{Q})$ and (ii) $(\mathcal{C}_{\mathcal{Q}}, \mathcal{K}_{\mathcal{Q}}, \mathcal{T}_{\mathcal{Q}}, \mathcal{R}_{\mathcal{Q}}) \subseteq^+ (\mathcal{C}_r, \mathcal{K}_r, \mathcal{T}_r, \mathcal{R}_r)$. In other words, if r belongs to $neg\text{-}sup\text{-}rules(\mathcal{Q})$. So,

$$\Delta\text{Cons}(\mathcal{Q}) = neg\text{-}sup\text{-}rules(\mathcal{Q}). \quad (2)$$

Definition 7. We say that a rule r (**partially**) supports a “yes” to \mathcal{Q} if (i) $\text{OP}_r = \text{OP}_{\mathcal{Q}}$, (ii) $(\mathcal{C}_{\mathcal{Q}}, \mathcal{K}_{\mathcal{Q}}, \mathcal{T}_{\mathcal{Q}}, \mathcal{R}_{\mathcal{Q}}) \subseteq^* (\mathcal{C}_r, \mathcal{K}_r, \mathcal{T}_r, \mathcal{R}_r)$ and (iii) $r \notin sup\text{-rule}(\mathcal{Q})$. Thus,

$$\partial\text{Pros}(\mathcal{Q}) = rel\text{-}rules(\mathcal{Q}) \setminus sup\text{-}rules(\mathcal{Q}). \quad (3)$$

Definition 8. We say that a rule r (**partially**) supports a “no” to \mathcal{Q} if (i) $\text{OP}_r \in neg(\mathcal{Q})$, (ii) $(\mathcal{C}_{\mathcal{Q}}, \mathcal{K}_{\mathcal{Q}}, \mathcal{T}_{\mathcal{Q}}, \mathcal{R}_{\mathcal{Q}}) \subseteq^* (\mathcal{C}_r, \mathcal{K}_r, \mathcal{T}_r, \mathcal{R}_r)$ and (iii) $r \notin neg\text{-}sup\text{-rule}(\mathcal{Q})$. Therefore,

$$\partial\text{Cons}(\mathcal{Q}) = neg\text{-}rel\text{-}rules(\mathcal{Q}) \setminus neg\text{-}sup\text{-}rules(\mathcal{Q}). \quad (4)$$

We use the SPARQL query of Listing 1.1 to retrieve elements of $sup\text{-}rules(\mathcal{Q})$. The WHERE clause of this SPARQL query strictly follows the expression given in Definition 1. In this SPARQL query, Class(OP) represents the class of the deontic operator OP, i.e., a sub-class of lrmlmm:DeoncticSpecification as shown in Fig. 1. Definition 6 gives two equivalent expressions of $neg\text{-}sup\text{-}rules(\mathcal{Q})$, where the second reuses $sup\text{-}rules(\mathcal{Q})$. Therefore we can adapt the SPARQL of Listing 1.1 with the suitable parameters, to build $neg\text{-}sup\text{-}rules(\mathcal{Q})$.

⁸ I.e., not the same and not broader than (within the meaning of \subseteq^+).

Listing 1.1. The SPARQL query to build $\text{sup-rules}(\mathcal{C}_Q, \mathcal{K}_Q, \mathcal{T}_Q, \text{OP}_Q, \mathcal{R}_Q)$

```
SELECT ?rule WHERE { ?rule a shom:Rule;
shom:ruleContext ?ctx; shom:ruleCondition ?cdt;
shom:ruleTarget ?tgt; shom:ruleRequirement ?req.
{C_Q rdfs:subClassOf* | a | owl:sameAs ?ctx} .
{K_Q rdfs:subClassOf* | a | owl:sameAs ?cdt} .
{T_Q rdfs:subClassOf* | a | owl:sameAs ?tgt} .
{?req a Class(OP_Q); shom:deonticAction ?act .
{R_Q rdfs:subClassOf* | a | owl:sameAs ?act} }
```

We rely on the SPARQL query of Listing 1.2 to build the set $\text{rel-rules}(Q)$. This query follows the expression of $\text{rel-rules}(Q)$ given in Definition 3. Also, following its second expression in Definition 4, we can retrieve elements of $\text{neg-rel-rules}(Q)$ using the SPARQL query of $\text{rel-rules}(Q)$.

Listing 1.2. The SPARQL query to build $\text{rel-rules}(\mathcal{C}_Q, \mathcal{K}_Q, \mathcal{T}_Q, \text{OP}_Q, \mathcal{R}_Q)$

```
SELECT ?rule WHERE { ?rule a shom:Rule;
shom:ruleContext ?ctx; shom:ruleCondition ?cdt;
shom:ruleTarget ?tgt; shom:ruleRequirement ?req.
{ {C_Q rdfs:subClassOf* | a | owl:sameAs ?ctx} UNION
{?ctx rdfs:subClassOf* | a C_Q} } UNION
{?subCtx rdfs:subClassOf* | a | owl:sameAs C_Q , ?ctx}} .
{ {K_Q rdfs:subClassOf* | a | owl:sameAs ?cdt} UNION
{?cdt rdfs:subClassOf* | a K_Q} } UNION
{?subCdt rdfs:subClassOf* | a | owl:sameAs K_Q , ?cdt}} .
{ {T_Q rdfs:subClassOf* | a | owl:sameAs ?tgt} UNION
{?tgt rdfs:subClassOf* | a T_Q} } UNION
{?subTgt rdfs:subClassOf* | a | owl:sameAs T_Q , ?tgt}} .
{ {?req a Class(OP_Q); shom:deonticAction ?act .
{R_Q rdfs:subClassOf* | a | owl:sameAs ?act} UNION
{?act rdfs:subClassOf* | a R_Q} } UNION
{?subAct rdfs:subClassOf* | a | owl:sameAs R_Q , ?act}}}} }
```

Having detailed how to obtain $\Delta\text{Pros}(Q)$, $\Delta\text{Cons}(Q)$, $\partial\text{Pros}(Q)$ and $\partial\text{Cons}(Q)$, we move to the next section where we present how to obtain exceptions to a conclusion.

5.3 Exceptions to a Conclusion

In some cases there are specific rules that counter-attack the conclusions $+C$, $-C$, $+c$ or $-c$. These rules are called “exceptions”. In Definition 9 below, C represents a conclusion, which can be $+C$, $-C$, $+c$ or $-c$.

Definition 9. An exception to a conclusion C is any rule r that (i) is partially in favor of an opposite conclusion C' to the one actually made (i.e., C), and (ii) is not overruled by a rule that fully supports the conclusion C .

Since an exception supports an opposite conclusion to the one actually made we must distinguish the case where the current conclusion is $+C$ or $+c$, and the case where it is $-C$ or $-c$. This is necessary because the criteria to be opposed to those two groups of conclusions are not the same:

1. A rule that counter-attacks $+C$ or $+c$ necessarily has a deontic operator that is in conflict with the one of the conclusion. Hence, in this case, a formal transcription of Definition 9 tells us that the set of exceptions equals to:

$$\left\{ r \in \overbrace{\bigcup_{\text{OP}' \in \text{neg}(\text{OP}_C)} \partial\text{Pros}(\mathcal{Q}_{C_C, K_C, T_C, \text{OP}', R_C})}^{\text{(i)}} \right. \\ \left. \wedge \underbrace{\#r' \in \Delta\text{Pros}(\mathcal{Q}_{C_C, K_C, T_C, \text{OP}_C, R_C}), r' > r}_{\text{(ii)}} \right\}. \quad (5)$$

In Eq. 5:

- Part (i) represents the formal expression of item (i) of Definition 9. There, we mention that a rule that is an exception to a “positive” conclusion (i.e., $+C$ or $+c$) necessarily has a partial support to a “yes” at any question with parameters related to those of the conclusion, *but with a conflicting deontic operator*.
 - Part (ii) represents the fact that the rule must not be overruled by another rule (fully) in favor of the conclusion (hence the use of ΔPros). It corresponds to item (ii) of Definition 9.
2. A rule that counter-attacks $-C$ or $-c$ must have a deontic operator that is the same as the one in the conclusion. Hence, similar to Eq. 5, a formal representation of Definition 9 tells us that, in this case, the set of exceptions equals to:

$$\left\{ r \in \overbrace{\partial\text{Pros}(C_C, K_C, T_C, \text{OP}_C, R_C)}^{\text{(i)}} \right. \\ \left. \wedge \#r' \in \Delta\text{Cons}(\mathcal{Q}_{C_C, K_C, T_C, \text{OP}_C, R_C}), r' > r \right\}. \quad (6)$$

It is important to notice that we must find exceptions to rules *recursively*. In other words, for rules that partially attack a conclusion, we must investigate whether these rules have exceptions in their turn, and so on.

In Fig. 2, we summarize the steps of our deontic reasoning process as it is implemented to answer a deontic question. In this figure:

- All steps before the dashed line map the process described in Sect. 5.1 where we identify the conclusion to the question.
- The processes labeled ① and ② correspond to Eqs. 5 and 6 respectively.

- In the process ❸, we recursively find exceptions, and then in ❹ we build the exceptions dependency graph \mathcal{G} . \mathcal{G} is oriented. Vertices of \mathcal{G} are the rules used to answer the question (i.e., ΔPros and ΔCons if non empty or ∂Pros and ∂Cons otherwise), including the exceptions identified during step ❸. Furthermore, (r, ρ) is an edge of \mathcal{G} if ρ is an exception of r . Hence, a cycle in \mathcal{G} is equivalent to an inconsistency.
- Step ❺ is a user friendly output of the answer.

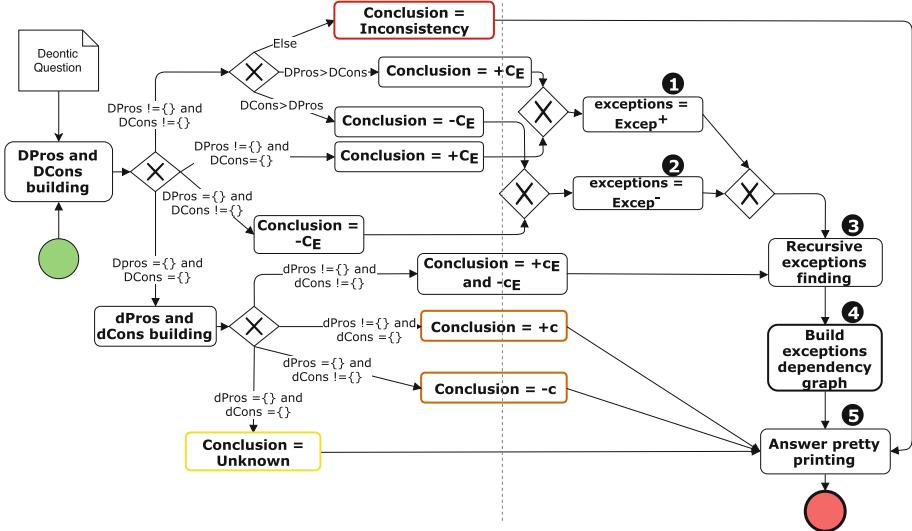


Fig. 2. Pipeline to answer a deontic question. (Color figure online)

Checking the consistency of a deontic rule base By analyzing the structure of the exceptions dependency graph we can now detect inconsistencies. But *this holds only for the subset of rules, including their exceptions, that are used to answer a given deontic question*. Therefore, the challenge is to ask a question Q_Ω that embraces all rules of the base, like: *Whatever the context, the condition and the target, is it permitted to do something?*. Formally, all the parameters of Q_Ω are equal to `owl:Thing`, so it is immediate to see that each rule of the base is related (in the meaning of the operator \subseteq^*). Consequently, according to Fig. 2, one of the following will always be true:

- We reach the red node so the inconsistency of the base is immediate.
- We reach one of the orange nodes, so there is no exception to a “yes” or “no” to Q_Ω and then no inconsistency is possible.
- We reach the yellow node, which means the rule base is empty, since parameters of any rule are necessarily related to those of Q_Ω , in which case we can conclude that the base is not inconsistent.

- We reach a conclusion which requires to find exceptions (i.e., $+C_E$, $-C_E$, $+c_E$ and $-c_E$), and knowing that the parameters of any rule are related to those of Q_Ω , the recursive search of exceptions will involve every rule of the base; having a cycle in this graph is equivalent to the presence of inconsistent requirements in the base.

6 Detailed Examples

We use the following excerpt of the French *arrêté préfectoral n° 96/2015⁹* to illustrate how we draw conclusions from a deontic rule base:

- **Article 1:** It is forbidden to navigate near the Paluel nuclear center (PNC).
- **Article 2:** In derogation to Article 1, are allowed to sail near the PNC: Military vessels; any vessel in need of assistance.

From these two articles, we extract the following rules and priority relations:

Listing 1.3. Example of deontic rules

```

1 :ps1 a lrmlmm:PrescriptiveStatement; lrmlmm:hasRule :R1.
2 :ps2 a lrmlmm:PrescriptiveStatement; lrmlmm:hasRule :R2, :R3.
3 :R1 a shom:Rule; shom:ruleTarget :Vessel; shom:ruleCondition owl:Thing ;
4 shom:ruleRequirement :ProhibitionNavPaluel ; shom:ruleContext owl:Thing ;
5 rdfs:comment "It is forbidden to navigate near the PNC" . #For any vessel
6 :R2 a shom:Rule; shom:ruleTarget :MilitaryVessel; shom:ruleCondition owl:Thing ;
7 shom:ruleRequirement :PermissionNavPaluel ; shom:ruleContext owl:Thing ;
8 rdfs:comment "Military vessels are authorised to navigate near the PNC" .
9 :R3 a shom:Rule; shom:ruleTarget :Vessel ; shom:ruleCondition owl:Thing ;
10 shom:ruleRequirement :PermissionNavPaluel ; shom:ruleContext :Assistance ;
11 rdfs:comment "Any vessel for assistance purpose is authorised to navigate..." .
12 :ProhibitionNavPaluel a lrmlmm:Prohibition; shom:deonticAction :NavPaluel.
13 :PermissionNavPaluel a lrmlmm:Permission; shom:deonticAction :NavPaluel.
14 :NavPaluel rdfs:subClassOf owl:Thing .
15 :Vessel rdfs:subClassOf owl:Thing .
16 :MilitaryVessel rdfs:subClassOf :Vessel .
17 :Assistance rdfs:subClassOf owl:Thing .
18 :PS2_OVER_PS1 a lrmlmm:Override; lrmlmm:over :ps2; lrmlmm:under :ps1.

```

Hence, for the three rules R1, R2 and R3 above, we have:

- | | | |
|--------------------------------|-------------------------------------|---------------------------------|
| – $C_{R1} = \text{owl:Thing}$ | – $C_{R2} = \text{owl:Thing}$ | – $C_{R3} = \text{:Assistance}$ |
| – $K_{R1} = \text{owl:Thing}$ | – $K_{R2} = \text{owl:Thing}$ | – $K_{R3} = \text{owl:Thing}$ |
| – $T_{R1} = \text{:Vessel}$ | – $T_{R2} = \text{:MilitaryVessel}$ | – $T_{R3} = \text{:Vessel}$ |
| – $OP_{R1} = \text{PRO}$ | – $OP_{R2} = \text{PER}$ | – $OP_{R3} = \text{PER}$ |
| – $R_{R1} = \text{:NavPaluel}$ | – $R_{R2} = \text{:NavPaluel}$ | – $R_{R3} = \text{:NavPaluel}$ |

⁹ The full (French) version of the text is available at <https://tinyurl.com/y77x32y3>.

The following relations between R1, R2 and R3 hold

$$(\mathcal{C}_{R2}, \mathcal{K}_{R2}, \mathcal{T}_{R2}, \mathcal{R}_{R2}) \subseteq^+ (\mathcal{C}_{R1}, \mathcal{K}_{R1}, \mathcal{T}_{R1}, \mathcal{R}_{R1}) \quad (7) \quad (\mathcal{C}_{R3}, \mathcal{K}_{R3}, \mathcal{T}_{R3}, \mathcal{R}_{R3}) \subseteq^+ (\mathcal{C}_{R1}, \mathcal{K}_{R1}, \mathcal{T}_{R1}, \mathcal{R}_{R1}) \quad (8)$$

$$(\mathcal{C}_{R2}, \mathcal{K}_{R2}, \mathcal{T}_{R2}, \mathcal{R}_{R2}) \subseteq^* (\mathcal{C}_{R1}, \mathcal{K}_{R1}, \mathcal{T}_{R1}, \mathcal{R}_{R1}) \quad (9) \quad (\mathcal{C}_{R3}, \mathcal{K}_{R3}, \mathcal{T}_{R3}, \mathcal{R}_{R3}) \subseteq^* (\mathcal{C}_{R1}, \mathcal{K}_{R1}, \mathcal{T}_{R1}, \mathcal{R}_{R1}) \quad (10)$$

Also, we recall that \subseteq^+ and \subseteq^* are *reflexive* relations.

6.1 \mathcal{Q}^a Is it Allowed to Navigate Near the PNC for Military Vessels?

We have: $\mathcal{C}_{\mathcal{Q}^a} = \text{owl:Thing}$, $\mathcal{K}_{\mathcal{Q}^a} = \text{owl:Thing}$, $\mathcal{T}_{\mathcal{Q}^a} = \text{:MilitaryVessel}$, $\text{OP}_{\mathcal{Q}^a} = \text{PER}$, and $\mathcal{R}_{\mathcal{Q}^a} = \text{:NavPaluel}$. We note that \mathcal{Q}^a and R2 have the same parameters, and the same deontic parameters, so Eqs. 7 and 9 above hold with \mathcal{Q}^a playing the role of R2. A rule ρ belongs to $\Delta\text{Pros}(\mathcal{Q}^a)$ if and only if $(\mathcal{C}_{\mathcal{Q}^a}, \mathcal{K}_{\mathcal{Q}^a}, \mathcal{T}_{\mathcal{Q}^a}, \mathcal{R}_{\mathcal{Q}^a}) \subseteq^+ (\mathcal{C}_\rho, \mathcal{K}_\rho, \mathcal{T}_\rho, \mathcal{R}_\rho)$ and $\text{OP}_{\mathcal{Q}^a} = \text{OP}_\rho$. So we have $\Delta\text{Pros}(\mathcal{Q}^a) = \{\text{R2}\}$. Also, $\gamma \in \Delta\text{Cons}(\mathcal{Q}^a)$ if and only if $(\mathcal{C}_{\mathcal{Q}^a}, \mathcal{K}_{\mathcal{Q}^a}, \mathcal{T}_{\mathcal{Q}^a}, \mathcal{R}_{\mathcal{Q}^a}) \subseteq^+ (\mathcal{C}_\gamma, \mathcal{K}_\gamma, \mathcal{T}_\gamma, \mathcal{R}_\gamma)$ and $\text{OP}_\gamma \in \text{neg}(\text{OP}_{\mathcal{Q}^a})$. So $\Delta\text{Cons}(\mathcal{Q}^a) = \{\text{R1}\}$ (thanks to Eq. 7). Now we must see whether one of $\Delta\text{Pros}(\mathcal{Q}^a)$ or $\Delta\text{Cons}(\mathcal{Q}^a)$ overrules the other. Line 18 of the Listing 1.3 implies $\Delta\text{Pros}(\mathcal{Q}^a) > \Delta\text{Cons}(\mathcal{Q}^a)$. So we conclude that we have $+ \mathfrak{C}_E$, i.e., *Yes, it is allowed to do so*. Next, we need to identify potential exceptions. Based on Eq. 5, we say that η is an exception to the above conclusion if $(\mathcal{C}_\eta, \mathcal{K}_\eta, \mathcal{T}_\eta, \mathcal{R}_\eta) \subseteq^* (\mathcal{C}_{\mathcal{Q}^a}, \mathcal{K}_{\mathcal{Q}^a}, \mathcal{T}_{\mathcal{Q}^a}, \mathcal{R}_{\mathcal{Q}^a})$, $(\mathcal{C}_{\mathcal{Q}^a}, \mathcal{K}_{\mathcal{Q}^a}, \mathcal{T}_{\mathcal{Q}^a}, \mathcal{R}_{\mathcal{Q}^a}) \not\subseteq^+ (\mathcal{C}_\eta, \mathcal{K}_\eta, \mathcal{T}_\eta, \mathcal{R}_\eta)$, and $\text{OP}_\eta \in \text{neg}(\text{OP}_{\mathcal{Q}^a})$. No rule in the base meets this criteria, so the answer to \mathcal{Q}^a remains unchanged (no exception added).

6.2 \mathcal{Q}^b Is Any Vessel Allowed to Navigate Near the PNC?

We have: $\mathcal{C}_{\mathcal{Q}^b} = \text{owl:Thing}$, $\mathcal{K}_{\mathcal{Q}^b} = \text{owl:Thing}$, $\mathcal{T}_{\mathcal{Q}^b} = \text{:Vessel}$, $\text{OP}_{\mathcal{Q}^b} = \text{PER}$, $\mathcal{R}_{\mathcal{Q}^b} = \text{:NavPaluel}$. We note that \mathcal{Q}^b and R1 have the same parameters *but have conflicting deontic operators*. $\Delta\text{Pros}(\mathcal{Q}^b) = \emptyset$ and $\Delta\text{Cons}(\mathcal{Q}^b) = \{\text{R1}\}$. Hence, we conclude $- \mathfrak{C}_E$ (see item 1c), i.e., *No, it is not allowed*. Let us now find exceptions. In this case, we refer to Eq. 6, so η is an exception to the above conclusion if $(\mathcal{C}_\eta, \mathcal{K}_\eta, \mathcal{T}_\eta, \mathcal{R}_\eta) \subseteq^* (\mathcal{C}_{\mathcal{Q}^b}, \mathcal{K}_{\mathcal{Q}^b}, \mathcal{T}_{\mathcal{Q}^b}, \mathcal{R}_{\mathcal{Q}^b})$, $(\mathcal{C}_{\mathcal{Q}^b}, \mathcal{K}_{\mathcal{Q}^b}, \mathcal{T}_{\mathcal{Q}^b}, \mathcal{R}_{\mathcal{Q}^b}) \not\subseteq^+ (\mathcal{C}_\eta, \mathcal{K}_\eta, \mathcal{T}_\eta, \mathcal{R}_\eta)$, and $\text{OP}_\eta = \text{OP}_{\mathcal{Q}^b}$. Thanks to Eqs. 7–10 we see that R2 and R3 meet these criteria. To be definitely considered as exceptions, they must not be overruled by any element of $\Delta\text{Cons}(\mathcal{Q}^b)$, i.e., R1. Actually this is the case, so they are exceptions to $- \mathfrak{C}_E$. A (recursive) search of exceptions to them remains unsuccessful. So the final answer to \mathcal{Q}^b is *No, it is not allowed to do so, but there are exceptions: – Military vessels are authorized to do so (Rule R2) – Any vessel in need of assistance does so (Rule R3)*.

6.3 \mathcal{Q}^c Is it Prohibited to Any Vessel to Navigate Near the PNC?

We use this question to illustrate partial conclusion. We assume for this case that the rule base contains only R2 and R3. We have: $\mathcal{C}_{\mathcal{Q}^c} = \text{owl:Thing}$, $\mathcal{K}_{\mathcal{Q}^c} = \text{owl:Thing}$, $\mathcal{T}_{\mathcal{Q}^c} = \text{:Vessel}$, $\mathcal{OP}_{\mathcal{Q}^c} = \text{PRO}$, $\mathcal{R}_{\mathcal{Q}^c} = \text{:NavPaluel}$. With R1 discarded from the base, we have $\Delta\text{Pros}(\mathcal{Q}^c) = \Delta\text{Cons}(\mathcal{Q}^c) = \emptyset$. Hence, we must find partial support and counter-attack, i.e., $\partial\text{Pros}(\mathcal{Q}^c)$ and $\partial\text{Cons}(\mathcal{Q}^c)$, to the assertion denoted by \mathcal{Q}^c . We have $\partial\text{Pros}(\mathcal{Q}^c) = \emptyset$ and $\partial\text{Cons}(\mathcal{Q}^c) = \{\text{R2, R3}\}$. Hence we conclude that we have $\neg c$, i.e., *No, it is not prohibited to do so, but rules in the base cover only specific parts of this case: – Military vessels are authorized to do so (Rule R2) – Any vessel in need of assistance is authorized to do so (Rule R3)*.

7 Future Work

As further developments of this work, we intend to perform the following tasks.

- **Broadening the scope of the rules.** As mentioned in Sect. 3, the scope of rules considered in this work is the one of prescriptive with not deontic formula in their body. We need to cover all types of prescriptive rules, i.e., also the case of constitutive rules. Hence, we must propose a modeling that take into account this extension. An interesting starting point for the modeling of constitutive rules would be the decompositions *The concept X counts as Y* or *The concept X counts as Y in context C* proposed by Searle [19].
- **Towards a proof theory.** We mentioned in the previous point the extension of the scope of rules. This implies updating the mechanisms of answering to deontic questions. Also we intend to propose a proof theory, i.e., how deontic and non deontic assertions are derived in a base made of constitutive and prescriptive rules.
- **Automatic question answering.** Currently we manually identify question parts (i.e., context, condition, target, deontic operator and requirement). It would be very useful to propose an approach to automatically decompose questions into their formal parts and then answer them automatically.

8 Conclusion

In this paper, we address the problem of deontic reasoning for legal ontologies. We tackle this issue in terms of deontic questions answering. First, we propose a new deontic rule modeling paradigm for prescriptive rules and then we use this modeling to present mechanisms for deontic question answering. The possible answers we provide to deontic questions, are as diverse as the ones given by a human, i.e., we are able to answer yes/no for the whole question or only for specific cases, with or without exceptions. We also provide answer unknown when there is no evidence in the rule base allowing to answer the question or unknown (potential inconsistencies) if they are unresolved conflicting rules. This work is a first step towards a proof theory for a deontic rule base.

Acknowledgements. This work is funded by the *Service hydrographique et océanographique de la marine* (Shom) as part of the REIZHMOR project.

References

1. Gordon, T.F.: The legal knowledge interchange format (LKIF). Technical report, ESTRELLA Project <http://www.estrellaproject.org/doc/Estrella-D4.1.pdf>
2. Boley, H., Tabet, S., Wagner, G.: Design rationale of RuleML: a markup language for semantic web rules. In: 1st International Conference on SW Working, pp. 381–401 (2001)
3. Palmirani, M., Governatori, G., Athan, T., Boley, H., Paschke, A., Wyner, A.: LegalRuleML core specification version 1.0. OASIS Committee Specification Draft 01 / Public Review Draft 01, October 2016
4. OMG: Semantics of Business Vocabulary and Business Rules (SBVR), v1.0. Technical report, Object Management Group (2008). <https://www.omg.org/spec/SBVR/1.0/>
5. Winkels, R., Boer, A., Hoekstra, R.: CLIME: lessons learned in legal information serving. In: Proceedings of the 15th ECAI, pp. 230–234. IOS Press (2002)
6. Valente, A., Breuker, J.: A functional ontology of law. Artif. Intell. law **7**, 241–361 (1994)
7. Gangemi, A.: Design patterns for legal ontology constructions. LOAIT **2007**, 65–85 (2007)
8. Lame, G.: Using NLP techniques to identify legal ontology components: concepts and relations. In: Benjamins, V.R., Casanovas, P., Breuker, J., Gangemi, A. (eds.) Law and the Semantic Web. LNCS (LNAI), vol. 3369, pp. 169–184. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-32253-5_11
9. Yurchyshyna, A., Zarli, A.: An ontology-based approach for formalisation and semantic organisation of conformance requirements in construction. Autom. Constr. **18**(8), 1084–1098 (2009)
10. Pauwels, P., et al.: A semantic rule checking environment for building performance checking. Autom. Constr. **20**(5), 506–518 (2011)
11. Kacfah Emani, C.: Automatic detection and semantic formalisation of business rules. In: Presutti, V., d'Amato, C., Gandon, F., d'Aquin, M., Staab, S., Tordai, A. (eds.) ESWC 2014. LNCS, vol. 8465, pp. 834–844. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-07443-6_57
12. Hassanpour, S., O'Connor, M.J., Das, A.K.: A framework for the automatic extraction of rules from online text. In: Bassiliades, N., Governatori, G., Paschke, A. (eds.) RuleML 2011. LNCS, vol. 6826, pp. 266–280. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22546-8_21
13. Kang, S., et al.: Extraction of manufacturing rules from unstructured text using a semantic framework. In: ASME 2015 American Society of Mechanical Engineers (2015)
14. Hoekstra, R., Breuker, J., Di Bello, M., Boer, A., et al.: The LKIF core ontology of basic legal concepts. LOAIT **321**, 43–63 (2007)
15. Gordon, T.F.: Constructing legal arguments with rules in the legal knowledge interchange format (LKIF). In: Casanovas, P., Sartor, G., Casellas, N., Rubino, R. (eds.) Computable Models of the Law. LNCS (LNAI), vol. 4884, pp. 162–184. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85569-9_11

16. Boley, H., Paschke, A., Shafiq, O.: RuleML 1.0: the overarching specification of web rules. In: Dean, M., Hall, J., Rotolo, A., Tabet, S. (eds.) RuleML 2010. LNCS, vol. 6403, pp. 162–178. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-16289-3_15
17. Van De Ven, S., Hoekstra, R., Breuker, J., Wortel, L., El Ali, A.: Judging amy: automated legal assessment using OWL 2. In: OWLED, vol. 432 (2008)
18. Antoniou, G., Billington, D., Governatori, G., Maher, M.J.: Representation results for defeasible logic. ACM TOCL **2**(2), 255–287 (2001)
19. Searle, J.R.: The Construction of Social Reality. Simon and Schuster, New York (1995)