Handling irresolvable conflicts in the Semantic Web: an RDF-based conflict-tolerant version of the Deontic Traditional Scheme

Livio Robaldo[1] and Gianluca Pozzato[2]

[1]School of Law, Swansea University
Singleton Park, Swansea, SA2 8PP, UK
livio.robaldo@swansea.ac.uk

[2]Department of Computer Science, University of Turin
Via Pessinetto, 12, Torino, 10149, Italy
gianluca.pozzato@unito.it

Abstract

This paper introduces a computational ontology for deontic reasoning, fully implemented in RDF* and SPARQL*, designed to support reasoning in the presence of irresolvable conflicts. These are situations in which two or more norms prescribe incompatible obligations, prohibitions, or permissions, without any clear priority among them. Existing approaches in formal deontic logic are typically limited to the propositional level, focused primarily on obligation as the central modality, and are rarely implemented in a way that is compatible with Semantic Web standards. The framework presented here addresses these limitations by providing a first-order, RDF-based formalization of all standard deontic modalities: obligations, permissions, optionality, and their negations. It supports the explicit representation and reasoning about violations and conflicts, while also accounting for contextual constraints. The ontology integrates contributions from three research areas that have so far largely developed in isolation: RDF-based LegalTech solutions, reification-based models of Natural Language Semantics, and conflict-tolerant approaches in formal deontic logic. By incorporating contradictions and conflicts into the object language, the ontology supports advanced reasoning tasks within a framework that adheres to W3C standards. This makes it suitable for integration into industrial LegalTech applications where normative reasoning is required[1].

[1]The research presented in this paper has received funding from The Alan Turing Institute UK-Italy Trustworthy AI Visiting Researcher Programme - https://www.turing.ac.uk/work-turing/uk-italy-trustworthy-ai-visiting-researcher-programme-0

1 Introduction

Normative reasoning aims to formalize norms from legislation using formal logic, enabling automated compliance checking and other legal reasoning tasks.

This paper focuses specifically on compliance checking over data encoded in the Resource Description Framework (RDF), the standard format for data interchange on the Web[2]. Given its growing adoption, RDF is likely to become the default knowledge representation format for symbolic AI more broadly, underscoring the need for systems that can perform compliance checking directly on RDF data (Robaldo et al., 2023).

(Robaldo et al., 2023) also emphasized the absence of a shared conceptualization of norms across existing systems, a limitation that hinders both the comparative evaluation of legal reasoners and the development of interoperable tools. Such comparisons are crucial to identify optimal solutions, yet they remain difficult without a common framework for representing norms. This lack of standardization also poses challenges for integrating components developed by heterogeneous sources into cohesive LegalTech solutions.

This paper aims to take a step toward filling that gap by proposing a novel framework implemented using Semantic Web technologies, i.e., a computational ontology for compliance checking designed for use within next-generation LegalTech applications.

Several approaches for compliance checking on RDF data have been proposed in the literature. Early works, such as (Gordon, 2008) and (Ceci, 2013), formalize norms using Semantic Web Rule Language (SWRL)[3] and Legal Knowledge Interchange Format (LKIF) rules (Hoekstra et al., 2007). More recent work, like (De Vos et al., 2019) and (Palmirani and Governatori, 2018), models norms using RuleML[4] and its legal extension, LegalRuleML[5]. Although LegalRuleML is aimed at standardizing legal norms, transformations are required to interpret these rules in executable reasoning languages.

Further developments, such as (Gandon, Governatori, and Villata, 2017), (Francesconi and Governatori, 2023), and (Robaldo, 2021), propose representing and reasoning with norms through SPARQL, OWL, and SHACL Triple rules, respectively. However, (Anim, Robaldo, and Wyner, 2024) recently pointed out that, on the one hand, OWL and SHACL Triple rules are limited in expressivity, particularly for complex reasoning involving aggregates and temporal data; on the other hand, representing defeasibility in SPARQL is challenging due to its lack of operators for rule prioritization. To address these issues, (Anim, Robaldo, and Wyner, 2024) use SHACL-SPARQL rules, which combine SHACL and SPARQL to offer greater expressive power.

All these approaches, along with those reviewed in (Robaldo et al., 2023), share a common foundation in deontic logic, which has served as the core formalism to model normative reasoning since the 1950s (Gabbay et al., 2013). In fact, the grounding in deontic logic is a defining feature of all contemporary approaches to legal reasoning.

On the other hand, we argue that insights from the literature on Natural Language Semantics must also be incorporated into LegalTech frameworks designed to process existing legislation. Since legislation is written in *natural language*, as intended for use and

[2]https://www.w3.org/RDF

[3]https://www.w3.org/Submission/SWRL; https://github.com/protegeproject/swrltab-plugin

[4]http://wiki.ruleml.org

[5]https://docs.oasis-open.org/legalruleml/legalruleml-core-spec/v1.0/os/legalruleml-core-spec-v1.0-os.html

interpretation by humans, the underlying logic must be capable of capturing its inherent nuances. Accordingly, we consider the integration of constructs from the literature in Natural Language Semantics essential for systems that aim to reason with legal norms as they are understood in *real-world* contexts.

A logical framework specifically designed to integrate deontic logic with Natural Language Semantics is reified Input/Output (I/O) logic (Robaldo et al., 2020) (Robaldo and Sun, 2017), primarily developed by the first author of this paper. Reified I/O logic incorporates the Natural Language Semantics framework developed by Jerry R. Hobbs over the course of his academic career, as detailed in (Gordon and Hobbs, 2017), into standard I/O logic (Makinson and van der Torre, 2000), a widely recognized framework that has mostly been used in past literature for modeling deontic logic.

This paper proposes a new framework as an *alternative* to reified I/O logic. While it retains the core insights of Hobbs' approach to Natural Language Semantics, it rejects the surrounding I/O logic axiomatization (as imported from (Sun and van der Torre, 2014) and (Parent and van der Torre, 2014)) for two main reasons.

First, reified I/O logic has been developed only at a theoretical level, and there is currently no automated reasoner capable of processing its formulae, even though (Sun and Robaldo, 2017) has shown that the logic is more tractable than standard deontic logics based on possible-world semantics.

Moreover, implementing such a reasoner using Semantic Web standards seems very challenging, which makes the first objective of this research, as outlined above, largely impractical.

Secondly, the axiomatization in (Sun and van der Torre, 2014) and (Parent and van der Torre, 2014) does not distinguish between logical contradictions and *irresolvable conflicts*, namely situations in which multiple obligations or other deontic statements apply simultaneously but cannot be all fulfilled.

Irresolvable conflicts, which will be defined, categorized, and exemplified in the next section, have been extensively studied in defeasible logic, argumentation theory, and AI more broadly. Conflicts are also central to Natural Language Semantics, which is inherently defeasible and often marked by conflicting interpretations. The meanings and inferences drawn from linguistic expressions are context-sensitive and can be overridden by additional information. This makes defeasibility a crucial feature of any formal system aiming to model or reason about natural language, especially in legal contexts.

On the other hand, because the distinction between contradictory and conflicting knowledge is not a core issue in traditional deontic logic, conflicts have received limited direct attention in that field, the main exceptions being the alternative deontic logics reviewed in (Goble, 2013), which we discuss further in Section 4 below.

In light of the prominent role of Natural Language Semantics in legal contexts, handling irresolvable conflicts is seen as a foundational requirement for any logical approach to reasoning in this domain and is therefore taken as the starting point for the RDF-based framework that will be defined in this paper.

The rest of the paper is organized as follows. The next section defines irresolvable conflicts in deontic logic from the perspective of legal reasoning. In particular, it presents the categorization of such conflicts developed by the jurist and legal philosopher Hans Kelsen, which this paper acknowledges. Section 3 introduces the main building block of the proposed computational ontology: the logical framework for Natural Language

Semantics developed by Jerry R. Hobbs, which we import from reified I/O logic. After that, Section 4 provides a brief review of the relevant literature in deontic logic, with a focus on the conflict-tolerant approaches surveyed in (Goble, 2013).

The subsequent three sections present the core contributions of this paper. Section 5 illustrates the implementation of Hobbs's framework, extended with deontic modalities, using Semantic Web standards. The proposed computational ontology specifically employs RDF* and SPARQL*, which are extensions of standard RDF and SPARQL that support a more natural and concise representation of metadata about statements[6]. Section 6 then extends this basic RDF*/SPARQL* implementation by incorporating rules to address all categories of irresolvable conflicts identified by Kelsen. Finally, Section 7 introduces additional constructs to properly model the interplay between deontic modalities, conflicts, and contextual constraints present in the state of affairs. A section devoted to future works and the conclusions close the paper.

All examples discussed in this paper, along with the Java code and detailed instructions for local execution, are available at: `https://github.com/liviorobaldo/conflict-tolerantDeonticTraditionalScheme`.

2 Irresolvable conflicts: Hans Kelsen's categorization

As explained in the Introduction, the proposed computational ontology is based on the notion of *conflicts* between deontic statements. Let us illustrate this notion by considering the following two obligations:

(1) a. It is obligatory to leave the building.

b. It is obligatory to not leave the building.

(1.a) could apply, for instance, in a situation where there is some danger *inside* the building, e.g., a fire in the building, while (1.b) could apply in a situation where there is some danger *outside* the building, e.g., a sandstorm.

The interesting question, of course, is what to do when there is *both* a fire inside the building *and* a sandstorm outside. In such a scenario, we must necessarily decide which of the two dangers we are willing to take the risk with, and accordingly violate the obligation associated with that danger.

As explained in the Introduction, most deontic logics proposed in the literature, including Standard Deontic Logic (von Wright, 1951) and (reified) Input/Output Logic (Sun and van der Torre, 2014), derive a contradiction (denoted by the symbol "$\perp$") from the conjunction of (1.a) and (1.b).

In our view, this approach is inadequate for modeling normative reasoning.

Contradictions should be associated with statements that are *illogical* in the state of affairs in which they are asserted, and, as such, always false. For example, the statement "Yoof is a dog and Yoof is a cat" is contradictory in a world where the sets of dogs and cats are disjoint. Such a statement would be illogical.

By contrast, the conjunction of (1.a) and (1.b) does not appear to be illogical, even in a scenario where there is both a fire inside the building and a sandstorm outside, a

[6]`https://www.w3.org/2022/08/rdf-star-wg-charter`

situation that could indeed occur in reality. In such a context, one of the two obligations will necessarily have to be violated. However, as is well known, violations are *not* contradictions. Therefore, there is no compelling reason to claim that a situation in which some obligations must be violated is itself contradictory, i.e., illogical.

In light of this, the framework proposed in this paper does not model (1.a) and (1.b) as contradictory, but rather as *conflicting* with one another, where a conflict is defined as "a situation in which two deontic statements hold in a given context, but complying with one of them entails violating the other". This definition, originally proposed by Hans Kelsen (Kelsen, 1991), also encompasses cases where one of the two deontic statements is a permission, although, as noted in (Vranes, 2006), such conflicts are said to be "unilateral", i.e., they exist in only one direction. A simple example is:

(2) a. It is prohibited to leave the building.

b. It is permitted to leave the building.

If both (2.a–b) hold and we choose not to leave the building in order to comply with (2.a), we cannot say that we are actually "violating" (2.b): the latter does not state that we *must* leave the building, i.e., it does not express an obligation. On the contrary, if we leave the building on the basis of what (2.b) authorizes us to do, we do violate (2.a).

A final category of conflicts discussed by Kelsen is exemplified in (3).

(3) a. It is obligatory to pay in cash.

b. It is obligatory to pay by card.

Kelsen classifies conflicts such as the one between (3.a) and (3.b) as "partial conflicts", because the conflict concerns only *part* of the same obligatory action: (3.a–b) prescribe two different instruments for paying, and the conflict arises from the fact that these instruments are mutually exclusive: payments made in cash are not made by card, and payments made by card are not made in cash.

In deontic logic literature, conflicts such as those exemplified in (1), (2), and (3) are also referred to as "irresolvable conflicts", where "irresolvable" means that neither of the two deontic statements is stronger than, or can override, the other. Several *conflict-tolerant* deontic logics have been proposed to formalize conflicts, though they mainly focus on conflicts between *obligations*, i.e., they do not address conflicts like the one exemplified in (2), where one of the two deontic statements is a permission.

Lou Goble is perhaps the author who has most thoroughly investigated the formalization of conflicts in deontic logic. His seminal work in (Goble, 2013) is still considered a reference survey[7] on the topic. The approaches reviewed in (Goble, 2013) will be briefly discussed below in Section 4.

However, at this stage, it is already worth noting that all conflict-tolerant deontic logics reviewed in (Goble, 2013) represent conflicts as *consistent* formulae; in this sense, these logics are said to be "conflict-tolerant". Most importantly, in almost[8] all of them, conflicts cannot be distinguished from other consistent formulae.

[7]See `https://plato.stanford.edu/entries/logic-deontic/#DeonDileConfToleDeonLogiRejeNC`

[8]The exceptions are Adaptive Deontic Logics, e.g., (Goble, 2014) (van De Putte, Beirlaen, and Meheus, 2019), where conflicts are viewed as "abnormalities" and stored in a separate set during the derivation. See subsubsection 4.1.3 below.

In our view, this is undesirable because, although conflicts are not contradictions, the logical framework must still be able to *explicitly represent* them, with the aim of *notifying* them, as they must eventually be *removed* from the normative system.

Indeed, it is not uncommon for existing legislation to contain conflicts among norms. These conflicts are rather difficult to identify manually, for instance, by the legislators responsible for updating the law. Artificial Intelligence (AI) could be of great help in this context, as it may enable the creation of LegalTech applications capable of *detecting* these conflicts, for the legislators to update the law so as to remove them.

Similar considerations can be found in a recent interview with Leon van der Torre and Dov Gabbay, published in (Steen and Benzmuller, 2024), in which the explicit representation of fallacies, violations, mistakes, etc. (henceforth referred to under the general term "abnormalities") necessary for *reasoning* about them has been identified as a crucial gap in contemporary logical frameworks for AI. Indeed, even in the LegalTech application envisioned above, if conflicts were explicitly represented, the application could not only notify them to the legislator but also *reason* about them, fit to suggest alternative solutions to resolve them, while assessing the pros and cons of each solution, etc., to help the legislator better ponder the decision on how to revise the law.

Furthermore, in our view, conflicts are not the only type of "abnormality" that ought to be notified. The state of affairs may also include *physical constraints* that either prevent compliance with obligations and prohibitions or prevent the execution of what is permitted. Situations like these must also be notified, because perhaps agents should not be sanctioned if it was *impossible* for them to comply with their obligations.

Contextual constraints might also be used to infer *how* they can comply with their obligations. For instance, consider the obligation in (4.a) and the constraint in (4.b).

(4) a. Whoever parks in a parking spot is obliged to pay £3 at the parking meter associated with that spot.

b. The parking meter in Sketty only accepts cash.

(4.b) states that, in Sketty, it is necessary to pay in cash. Therefore, if John is parking in Sketty, he will infer not only that he is obliged to pay £3, but more specifically, that he is obliged to pay the £3 *in cash*.

Finally, contextual constraints might also interact with conflicts. For instance, suppose that in the future the government decides to abolish cash as a way to combat corruption. In this new context, only payments by digital means, e.g., credit card, are allowed. The following prohibition is then added to the normative system:

(5) It is prohibited to pay in cash.

Now, it becomes impossible for John to comply with both (4.a) and (5) when he parks in Sketty: the physical constraint in (4.b) prevents compliance with the prohibition in (5). So, either John will violate this prohibition, or he will violate his obligation to pay £3. In either case, John's violation appears to be justified, suggesting that he perhaps should not be sanctioned for it.

The interplay between norms and contextual constraints has been scarcely addressed in past conflict-tolerant deontic logics, at least not with the level of granularity exemplified in (4) and (5). Most deontic logics proposed in the literature use *propositional* symbols as atomic formulae. However, to model examples like (4) and (5), we need a first-order framework capable of distinguishing between actions (e.g., paying) and their thematic roles (e.g., the *instrument* of paying, cash rather than card).

This paper presents a novel computational ontology to represent and reason with conflicts between deontic statements in all the cases exemplified so far, most of which are not currently addressed by state-of-the-art conflict-tolerant deontic logics. The proposed ontology aims to be a first step toward the future development of LegalTech applications capable of detecting and reasoning with conflicts between norms in legislation.

The next two sections respectively illustrate the two main foundations from the literature that inform the computational ontology proposed in this paper: the logic for Natural Language Semantics developed by Jerry R. Hobbs throughout his academic career, and the key developments in past deontic logic literature, with particular attention to the conflict-tolerant systems reviewed in (Goble, 2013).

The subsequent sections then build on the main insights from these two areas of research to define a framework for representing and reasoning with conflicts, as well as their interplay with constraints present in the state of affairs.

3 Background: encoding natural language statements in Jerry R. Hobbs's framework

This section introduces the framework outlined in (Gordon and Hobbs, 2017), along with earlier foundational work by Jerry R. Hobbs, particularly (Hobbs, 2003). This framework forms the basis for the computational ontology proposed in this paper.

(Gordon and Hobbs, 2017) presents a first-order logical framework for natural language semantics that is heavily based on the notion of *reification*. Philosophically, reification refers to the process of treating abstract entities as concrete objects in the world. In formal logic, this corresponds to representing such entities as first-order individuals, i.e., as constants or variables within the logic.

The concept of reification was first introduced by Donald Davidson in (Davidson, 1967), though his use of reification was limited to a narrow set of abstract entities. In contrast, (Gordon and Hobbs, 2017) adopts a massive reification approach, whereby any abstract entity may be reified into a first-order individual. Once reified, further assertions can be made about these individuals, and those assertions themselves can, in turn, be reified again into new individuals, allowing for recursive layers of abstraction and analysis.

According to (Gordon and Hobbs, 2017) and its philosophical and psycholinguistic foundations, reification reflects the way people naturally conceptualize events, actions, and states, and express them in language. For this reason, Hobbs's framework is particularly well-suited to modeling the semantics of natural language expressions, and it has been adopted here as the foundational logic for the proposed computational ontology.

Let's illustrate how reification works on a simple example: the assertion "John leaves". In standard first-order logic, this would typically be represented as `leave(John)`, where `leave` is a first-order predicate and `John` is a first-order individual. In contrast, (Gordon and Hobbs, 2017) associates this assertion with a distinct first-order individual `elj`, referred to as an "eventuality". According to the terminology used in (Gordon and Hobbs, 2017), the assertion is said to be "reified into" the eventuality `elj`. In this paper, we will also use the phrase "the fact that" to describe the relationship between eventualities and the corresponding assertions; thus, `elj` denotes *the fact that* John leaves.

Since eventualities are treated as first-order individuals, they can be used as arguments of first-order predicates. In particular, eventualities that reify actions or states such as "to leave" may serve as arguments for predicates expressing specific *modalities* applicable to those actions or states. In light of this, (Gordon and Hobbs, 2017) reformulates `leave`(`John`) into the following formalization:

$$\texttt{Rexist}(\texttt{elj}) \wedge \texttt{leave'}(\texttt{elj}, \texttt{John})$$

In this representation, the predicate `leave` is transformed into the primed predicate `leave'`, which includes an additional argument corresponding to the reified action. This reified individual is then used as the argument of another predicate, `Rexist`, which captures the *modality* associated with John's act of leaving. Specifically, `Rexist`(`elj`) asserts that `elj` *really exists* in the actual state of affairs.

The use of the predicate `Rexist` highlights a fundamental difference between how meaning is typically represented in standard first-order logic, e.g., `leave`(`John`), and how it is modeled in the framework proposed in (Gordon and Hobbs, 2017). In this framework, actions are not considered true or false for given individuals; rather, they are treated as eventualities that may or may not occur in the real world.

However, `Rexist` is not the only available modality. As (Gordon and Hobbs, 2017) explains, an eventuality may be part of someone's beliefs without actually occurring in the real world, or it may exist within a fictional context, or be merely possible or probable rather than real. Each of these alternative modalities must be represented by a unary predicate different from `Rexist`. This paper will later introduce predicates to represent deontic modalities, which are necessary for modeling compliance checking, as discussed above in the Introduction.

Furthermore, to facilitate the encoding of the formulae in (Gordon and Hobbs, 2017) into RDF, we will adopt the specific pattern proposed in (Robaldo et al., 2023), shown in (6), rather than defining primed predicates derived from their unprimed counterparts. This basic pattern will later be implemented and extended using RDF* and SPARQL*.

In (6), `e` denotes an eventuality representing either an action or a state, as specified by the predicate `ET`, which stands for "Eventuality Type". The predicate `M` denotes the modality of the eventuality. Finally, $\texttt{t}_\texttt{i}$ represents thematic roles, while $\texttt{v}_\texttt{i}$ denotes their corresponding values.

(6) $\texttt{M(e)} \wedge \texttt{ET(e)} \wedge \bigwedge_{i=1}^{n} \texttt{t}_\texttt{i}\texttt{(e, v}_\texttt{i}\texttt{)}$

In this pattern, the assertion "John leaves" is represented as shown in (7.a). Another example is provided in (7.b), which represents the assertion "John pays £3 in cash".

(7) a. `Rexist(elj)` $\wedge$ `Leave(elj)` $\wedge$ `has-agent(elj, John)`

b. `Rexist(epj)` $\wedge$ `Pay(epj)` $\wedge$ `has-agent(epj, John)` $\wedge$
`has-object(epj, £3)` $\wedge$ `has-instrument(epj, cash)`

In (7.a), `Leave` is a predicate distinct from the previous `leave` predicate, as it applies to an eventuality rather than to a person. Therefore, while `leave(John)` indicates that `John` belongs to the set of leavers, `Leave(elj)` indicates that `elj` belongs to the set of leaving actions. However, `Leave(elj)` does not state whether `elj` really exists in the state of affairs or only in someone's imagination, etc. To assert that `elj` really exists in the state of affairs, `Rexist(elj)` must be stated, thereby making `elj` belong to the set of really existing eventualities.

The framework in (Gordon and Hobbs, 2017) encompasses a broad theory of commonsense psychology, formalizing sets, abstract and instantiated eventualities, causal and temporal reasoning, composite entities, defeasibility, and more, all via reification. The objectives of this paper do not require all these concepts; therefore, only the predicates and axioms from (Gordon and Hobbs, 2017) that are essential for implementing the proposed compliance checking framework will be imported.

3.1 Negation, conjunction, and disjunction of eventualities

(Gordon and Hobbs, 2017) defines the three predicates `not`, `and`, and `or` to relate pairs or triples of eventualities. The predicates `not`, `and`, and `or` are intuitively defined as follows:

(8)

a. `not(en, e)`, where `en` and `e` are eventualities, states that `en` and `e` are *opposite* eventualities. For example, if `e` refers to the fact that "John leaves", `en` refers to the fact that "John does not leave".

b. `and(ea, e1, e2)` states that `ea` represents the *co-occurrence* of `e1` and `e2`; for example, if `e1` and `e2` refer respectively to the facts that "John eats" and "John drinks", then `ea` may represent the fact that "John eats and drinks".

c. `or(eo, e1, e2)` states that `eo` represents the *disjunctive occurrence* of `e1` and `e2`; for example, if `e1` and `e2` refer respectively to the facts that "John eats" and "John drinks", then `eo` may represent the fact that "John eats or drinks".

It is important to understand that these predicates do *not* correspond to the standard Boolean connectives ¬, ∧, and ∨. The latter operate on entire formulae, producing new formulae whose truth values depend on those of the initial formulae. In contrast, the former are predicates that relate eventualities, thereby forming atomic formulae.

The connectives ¬, ∧, and ∨ are used, however, to constrain the meaning of `not`, `and`, and `or` in relation to the various modalities. For the modality `Rexist`, the bi-implications in (9) are stipulated as valid.

(9)

a. $\forall_{\mathtt{e,\,en}}[\mathtt{not(en,\,e)}\rightarrow(\mathtt{Rexist(e)}\leftrightarrow\neg\mathtt{Rexist(en)})]$

b. $\forall_{\mathtt{ea,\,e1,\,e2}}[\mathtt{and(ea,\,e1,\,e2)}\rightarrow(\mathtt{Rexist(ea)}\leftrightarrow(\mathtt{Rexist(e1)}\wedge\mathtt{Rexist(e2)}))]$

c. $\forall_{\mathtt{ea,\,e1,\,e2}}[\mathtt{or(ea,\,e1,\,e2)}\rightarrow(\mathtt{Rexist(ea)}\leftrightarrow(\mathtt{Rexist(e1)}\vee\mathtt{Rexist(e2)}))]$

Therefore, if the fact that John leaves really exists, then it is false that the fact that John does not leave really exists (and vice versa). Similarly, if the fact that John eats and drinks really exists, then both the fact that John eats and the fact that John drinks really exist (and vice versa). Finally, if the fact that John eats or drinks really exists, then either the fact that John eats or the fact that John drinks really exists (or both), although we cannot necessarily determine which one, or whether both, really exist.

However, the bi-implications in (9) apply only to the `Rexist` modality and do not extend to the deontic modalities, for which additional axioms will be provided below.

The bi-implications in (9) ensure that any theorem involving $\neg$, $\wedge$, and $\vee$ applied to the modality `Rexist` is equivalent to the corresponding theorem using the predicates `not`, `and`, and `or`. For example, the formula in (10.a) bi-implicates the formula in (10.b), as per the equivalences in (9). Both (10.a) and (10.b) instantiate the so-called "Disjunctive Syllogism", i.e., the implication $(\mathtt{A} \vee \mathtt{B}) \wedge \neg\mathtt{B} \rightarrow \mathtt{A}$, on the predicate `Rexist`.

(10)
a. $\forall_{\mathtt{eo,e1,e2,en1}}[(\mathtt{Rexist(eo)} \wedge \mathtt{or(eo, e1, e2)} \wedge \mathtt{Rexist(en1)} \wedge \mathtt{not(en1, e1)}) \rightarrow \mathtt{Rexist(e2)}]$

b. $\forall_{\mathtt{e1,\,e2}}[((\mathtt{Rexist(e1)} \vee \mathtt{Rexist(e2)}) \wedge \neg\mathtt{Rexist(e1)}) \rightarrow \mathtt{Rexist(e2)}]$

Other theorems on the modality `Rexist`, such as the distributivity laws for conjunction and disjunction or De Morgan's laws, could similarly be "translated" between the predicates `not`, `and`, and `or` and the boolean connectives $\neg$, $\vee$, and $\wedge$. However, these are not included in the formalization below, as they are not relevant to the objectives of this paper. Instead, this paper will focus on formalizing deontic modalities, for which, as explained above, similar equivalences between the predicates `not`, `and`, and `or` and the Boolean connectives $\neg$, $\wedge$, and $\vee$ do not hold.

3.2 Abstract eventualities and their instantiations

The second main notion imported from (Gordon and Hobbs, 2017) is the distinction between abstract eventualities and their instantiations. Understanding this distinction is crucial for determining when contradictions or conflicts may arise. Consider:

(11)
a. John does not pay.

b. John pays £3 in cash.

It seems unquestionable that the two sentences in (11.a) and (11.b) contradict each other: if John makes no payment, then he cannot be paying £3 in cash. The same considerations apply to the following pair of sentences:

(12)
a. John pays in cash.

b. John pays by card.

Nevertheless, the contradictions between the sentences in (11) and (12) arise only under a crucial assumption, namely that the sentences are made within the same *context*, meaning they are both understood to apply "now" and thus refer to the same act of paying.

When we read the two sentences one after the other, we implicitly assume that they do; however, this assumption is actually a *pragmatic implicature*[9], which must be explicitly enforced in the derivations, in order to derive the contradiction.

In Hobbs's logic, this is also formally required because each sentence in (11) and (12) corresponds to different eventualities. For example, using the pattern in (6), sentences (11.a-b) could be respectively represented as follows:

(13)
a. `Rexist(enpja)` $\wedge$ `not(enpja, epja)` $\wedge$ `Pay(epja)` $\wedge$ `has-agent(epja, John)`

b. `Rexist(epj)` $\wedge$ `Pay(epj)` $\wedge$ `has-agent(epj, John)` $\wedge$
`has-object(epj, £3)` $\wedge$ `has-instrument(epj, cash)`

(13.a) involves two opposite eventualities: `enpja` and `epja`, while (13.b) contains a single eventuality: `epj`. Notably, `enpja` and `epja` denote a *cluster* of eventualities, unlike `epj`, which refers to a *single* eventuality. This is because if John does not pay, it implies he did not pay *anything* (neither £3, nor £4, etc.) using *any* instrument (cash, card, etc.).

How, then, can we represent and infer the *relationship* between `epja` and `epj` in a way that allows us to derive a contradiction between the two statements they represent?

To address representational challenges like these, Hobbs distinguishes between *abstract eventualities* and their *instantiations*. The former are generic statements that hold without necessarily requiring physical presence or concrete realization. These abstract eventualities can then be instantiated into less abstract eventualities (partial instantiations) or even into concrete eventualities (full instantiations). Instantiations represent one of the different "variants" through which the eventuality can become real.

In light of this, asserting that an abstract eventuality does not exist entails that none of its (partial or full) instantiations exist. Conversely, asserting that an abstract eventuality exists entails that at least one of its partial instantiations and at least one of its full instantiations exists. This is what will specifically enable the proposed computational ontology to derive contradictions or conflicts.

This paper adopts the conceptual distinction between abstract eventualities and their instantiations from (Gordon and Hobbs, 2017), but it uses a different, and hopefully clearer[10], formalization.

Abstract eventualities and their full instantiations (referred to as "instances", following (Gordon and Hobbs, 2017)) are respectively represented by the unary predicates `AbstractEventuality` and `Instance`, which are disjoint. Both predicates specialize the predicate `Eventuality`. While only instances can satisfy the `Rexist` modality, a separate

[9] See `https://plato.stanford.edu/entries/implicature`

[10] (Gordon and Hobbs, 2017) distinguishes between partial and full instantiations using two different predicates: `partialInstance` and `instance`. The former refines abstract eventualities into less abstract ones, while the latter refines abstract eventualities into their full instantiations. Conversely, (Gordon and Hobbs, 2017) defines a single unary predicate, `Eventuality`, that holds true for all eventualities. Here we prefer to take the opposite approach, i.e., to distinguish between abstract eventualities and instances at the class/predicate level. For this reason, this paper defines a single binary predicate, `instantiates`, along with two unary predicates: `AbstractEventuality` and `Instance`. Moreover, (Gordon and Hobbs, 2017) also applies the `Rexist` modality to abstract eventualities, while we now believe that stating that *abstract* eventualities *really* exist sounds somewhat awkward; therefore, we prefer to further distinguish abstract eventualities from instances using two different modalities: `Subsist` and `Rexist`.

modality is introduced for abstract eventualities, denoted by the predicate `Subsist`. This distinction is absent in (Gordon and Hobbs, 2017), where `Rexist` is used for both abstract eventualities and their instantiations. `Subsist` denotes a form of existence in abstraction, i.e., something that "exists" but does not necessarily have a material presence.

The proposed computational ontology stipulates that the `Subsist` modality applies exclusively to abstract eventualities, while the `Rexist` modality applies exclusively to instances. Consequently, when either of these modalities is applied to an eventuality, the corresponding subcategory of that eventuality can be inferred. The predicates `not`, `and`, and `or`, introduced in the previous section, apply to the `Subsist` modality in the same manner as they do to `Rexist` (simply replace `Rexist` with `Subsist` in (9)). Moreover, if one of the eventualities involved in a `not`, `and`, or `or` expression is an abstract eventuality (or an instance), then the other must be as well.

A binary predicate `instantiates` is then introduced to relate abstract eventualities to their instantiations (whether partial or full). `instantiates` denotes a reflexive and transitive relation; therefore, every abstract eventuality instantiates itself, and if `e1` instantiates `e2`, and `e2` instantiates `e3`, then `e1` also instantiates `e3`.

Additionally, the proposed computational ontology includes the inference rule in (14) to derive that if an eventuality is an action or state of the same type as another (abstract) eventuality, and the former specifies a superset of thematic roles with identical values for the shared ones, then the former `instantiates` the latter. In (14), the two "$\neg\exists$" conditions in the antecedent of the implication assert that there must be no thematic role `tr` specified for the abstract eventuality `ea` that is missing from its instantiation `ei`, and no thematic role `tr` specified for both with differing values, `va` and `vi`.

(14) $\forall_{\mathtt{ea,ei,ET}}[(\mathtt{AbstractEventuality(ea)}\wedge\mathtt{ET(ea)}\wedge\mathtt{ET(ei)}\wedge$
$\neg\exists_{\mathtt{tr,va}}[\mathtt{tr(ea,\ va)}\wedge\neg\exists_{\mathtt{vi}}[\mathtt{tr(ei,\ vi)}]]\wedge$
$\neg\exists_{\mathtt{tr,va,vi}}[\mathtt{tr(ea,\ va)}\wedge\mathtt{tr(ei,\ vi)}\wedge(\mathtt{va}\neq\mathtt{vi})])\rightarrow\mathtt{instantiates(ei,\ ea)}]$

(14) corresponds to the pragmatic implicature mentioned above. In other words, it allows one to assume that the statements in this paper's examples refer to the same actions or states whenever they are presented together. Naturally, this assumption does not always hold, and therefore, the implication in (14) is not universally valid.

Now, as Hobbs explicitly states, what is typically negated in NL statements like (11.a) is an abstract eventuality. Therefore, given the new modality `Subsist`, (11.a) above is not represented as in (13.a), but rather as in (15.a). On the other hand, the formalization of (11.b) remains unchanged and is repeated again in (15.b) for the reader's convenience.

(15)
a. $\mathtt{Subsist(enpja)}\wedge\mathtt{not(enpja,\ epja)}\wedge\mathtt{Pay(epja)}\wedge\mathtt{has\text{-}agent(epja,\ John)}$

b. $\mathtt{Rexist(epj)}\wedge\mathtt{Pay(epj)}\wedge\mathtt{has\text{-}agent(epj,\ John)}\wedge$
$\mathtt{has\text{-}object(epj,\ £3)}\wedge\mathtt{has\text{-}instrument(epj,\ cash)}$

Since (15.a-b) are uttered in the same context, the pragmatic implicature in (14) allows us to derive that `epj` instantiates `epja`, i.e., that `instantiates(epj, epja)` holds. In parallel, $\neg$`Subsist(epja)` is derived from (15.a) via the axioms previously introduced for the predicate not. Finally, given that, as explained above, the non-subsistence of an abstract eventuality entails the non-existence of all its full instantiations, $\neg$Subsist(epja) and Rexist(epj) are inferred to be contradictory.

On the other hand, the contradiction between the two fully instantiated eventualities corresponding to (12.a) and (12.b) arises from the assumption that they instantiate the same abstract eventuality, i.e., that they refer to the same (abstract) payment made by John in the state of affairs.

To derive that, a second implicature is introduced in (16): if the state of affairs includes two eventualities of the same type, under the same modality, and performed by the same agent, then (16) allows us to derive the existence of an abstract eventuality that both instantiate.

(16) $\forall_{\mathtt{ei1,ei2,ET,a}}[(\mathtt{ET(ei1)} \wedge \mathtt{ET(ei2)} \wedge \mathtt{M(ei1)} \wedge \mathtt{M(ei2)} \wedge \mathtt{has\text{-}agent(ei1, a)} \wedge \mathtt{has\text{-}agent(ei2, a)}) \rightarrow \exists_{\mathtt{ea}}[\mathtt{instantiates(ei1, ea)} \wedge \mathtt{instantiates(ei2, ea)}]]$

Once again, the implicature in (16) cannot be considered universally valid, i.e., in every possible state of affairs. This implicature (and the one in (14)) is appropriate in the present context because we consider only actions occurring "here" and "now", focusing on those that a single person cannot perform simultaneously. Under these constraints, both implicatures appear reasonable. However, in a more general setting, e.g., one that also involves temporal aspects, as outlined in the future work section below, it becomes necessary to introduce more sophisticated implicatures to determine when an abstract eventuality instantiates another eventuality in context, or when two eventualities are both instantiations of the same abstract eventuality.

The proposed computational ontology then stipulates that if two instantiations of the same abstract eventuality share at least one thematic role, and the values of those roles are mutually exclusive, the two instantiations cannot coexist.

To enforce this in the example in (12), we add the two implications in (17), which state that card and cash are mutually exclusive instruments for payment actions (i.e., if a payment e was made in cash, then it was not made by card, and vice versa), as well as the implication in (18), which states that if two eventualities instantiate the same abstract eventuality but differ in at least one thematic role, where it is asserted that one eventuality has a certain value for that role and the other does not, then it is inferred that the two eventualities are linked by the predicate not: if one eventuality really exists, the other one does not, and vice versa.

(17)

a. $\forall_{\mathtt{e}}[(\mathtt{Pay(e)} \wedge \mathtt{has\text{-}instrument(e, cash)}) \rightarrow \neg\mathtt{has\text{-}instrument(e, card)}]$

b. $\forall_{\mathtt{e}}[(\mathtt{Pay(e)} \wedge \mathtt{has\text{-}instrument(e, card)}) \rightarrow \neg\mathtt{has\text{-}instrument(e, cash)}]$

(18) $\forall_{\mathtt{e1,e2,ea,TR,v}}[(\mathtt{instantiates(e1, ea)} \wedge \mathtt{instantiates(e2, ea)} \wedge \mathtt{TR(e1, v)} \wedge \neg\mathtt{TR(e2, v)}) \rightarrow \mathtt{not(e1, e2)}]$

The above implications correctly lead to the inference that the logical representations of (12.a) and (12.b), shown respectively in (19.a) and (19.b), are inconsistent.

(19)
a. $\texttt{Rexist(e1)} \wedge \texttt{Pay(e1)} \wedge \texttt{has-agent(e1, John)} \wedge \texttt{has-instrument(e1, cash)}$

b. $\texttt{Rexist(e2)} \wedge \texttt{Pay(e2)} \wedge \texttt{has-agent(e2, John)} \wedge \texttt{has-instrument(e2, card)}$

The two implications in (17) entail that the instrument of `e1` is not `card`, and that the instrument of `e2` is not `cash`:

(20) $\neg\texttt{has-instrument(e1, card)} \wedge \neg\texttt{has-instrument(e2, cash)}$

From (20), the implication in (18) entails `not(e1, e2)` and, symmetrically, `not(e2, e1)`. From these, the axiom shown above in (9.a) symmetrically derive $\neg$`Rexist(e1)` and $\neg$`Rexist(e2)`, which contradict the corresponding positive literals in (19.a) and (19.b).

4 The Deontic Traditional Scheme and the state-of-the-art conflict-tolerant deontic logics

Early studies in deontic logic can be traced back to the Middle Ages (Knuutila, 1981). The Stanford Encyclopedia of Philosophy[11] provides a comprehensive literature review of various systems of deontic logic, based on (McNamara, 1996a) and (McNamara, 1996b).

Six normative statuses have been historically identified and are widely accepted by the contemporary scientific community: obligatory (`OB`), permitted (`PE`), prohibited[12] (`PR`), omissible (`OM`), optional (`OP`), and non-optional (`NO`). In formal deontic logic, these statuses are typically expressed as *deontic modalities* applied to a well-formed formula of the underlying logic, e.g., a proposition p in the case of propositional logic. Thus, `OB(p)`, `PE(p)`, `PR(p)`, `OM(p)`, `OP(p)`, and `NO(p)` are basic deontic (propositional) statements indicating that the proposition p is, respectively, obligatory, permitted, prohibited, omissible, optional, and non-optional.

It is also widely acknowledged that the six deontic modalities are logically interconnected. In particular, `OB` has traditionally been regarded as the primary deontic modality, from which the other five can be formally derived, as shown in (21). The Stanford Encyclopedia of Philosophy refers to (21) as "The Traditional Definitional Scheme".

(21)
(a) $\texttt{PE(p)} \leftrightarrow \neg\texttt{OB(}\neg\texttt{p)}$

(b) $\texttt{PR(p)} \leftrightarrow \texttt{OB(}\neg\texttt{p)}$

(c) $\texttt{OM(p)} \leftrightarrow \neg\texttt{OB(p)}$

(d) $\texttt{OP(p)} \leftrightarrow (\neg\texttt{OB(p)} \wedge \neg\texttt{OB(}\neg\texttt{p))}$

(e) $\texttt{NO(p)} \leftrightarrow (\texttt{OB(p)} \vee \texttt{OB(}\neg\texttt{p))}$

[11] `https://plato.stanford.edu/entries/logic-deontic`

[12] The Stanford Encyclopedia of Philosophy uses the term "impermissible" instead of "prohibited"; however, we prefer the latter as it is more commonly used in contemporary everyday language. For our purposes, the two terms are considered equivalent.

Moreover, the deontic modalities are logically related to one another, as illustrated in the Deontic Hexagon[13], shown in Figure 1, which is also referred to in the Stanford Encyclopedia of Philosophy as the "Traditional Threefold Classification".

This paper takes into account both the logical entailments defined in the Traditional Definitional Scheme (i.e., those in (21)) and those depicted in the Traditional Threefold Classification (i.e., the Deontic Hexagon). The union of these entailments is hereafter referred to as the "Deontic Traditional Scheme".

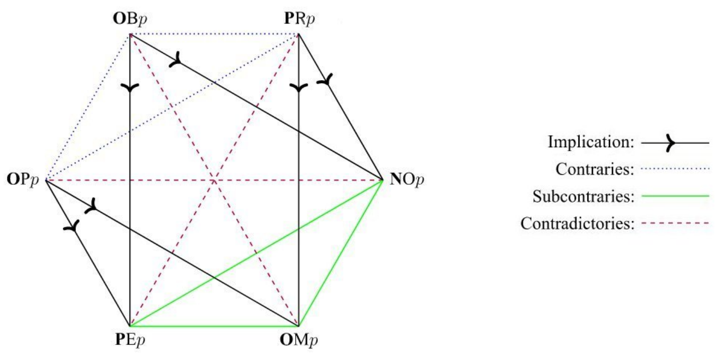

Figure 1: The Deontic Hexagon. In the hexagon, when two vertices are connected by an "Implication" arrow, the antecedent logically entails the consequent; when marked as "Contraries", both cannot be true; when marked as "Subcontraries", both cannot be false; and when marked as "Contradictories", they always have opposing truth values, i.e., they can neither both be true nor both be false.

[13] Figure 1 is taken from `https://plato.stanford.edu/entries/logic-deontic/#TradScheModaAnal`; in line with the previous footnote, we have replaced the label "IMp" (standing for "impermissible") with "PRp" (standing for "prohibited").

In symbols, as explained in the caption of Figure 1, the Deontic Hexagon specifies that:

(22) i. Implication:
  (a) $\texttt{OB(p)} \rightarrow \texttt{PE(p)}$
  (b) $\texttt{OB(p)} \rightarrow \texttt{NO(p)}$
  (c) $\texttt{PR(p)} \rightarrow \texttt{OM(p)}$
  (d) $\texttt{PR(p)} \rightarrow \texttt{NO(p)}$
  (e) $\texttt{OP(p)} \rightarrow \texttt{PE(p)}$
  (f) $\texttt{OP(p)} \rightarrow \texttt{OM(p)}$

ii. Contraries:
  (a) $\neg\texttt{(OB(p)} \wedge \texttt{PR(p))}$ equivalent to $\texttt{OB(p)} \rightarrow \neg\texttt{PR(p)} \wedge \texttt{PR(p)} \rightarrow \neg\texttt{OB(p)}$
  (b) $\neg\texttt{(OB(p)} \wedge \texttt{OP(p))}$ equivalent to $\texttt{OB(p)} \rightarrow \neg\texttt{OP(p)} \wedge \texttt{OP(p)} \rightarrow \neg\texttt{OB(p)}$
  (c) $\neg\texttt{(PR(p)} \wedge \texttt{OP(p))}$ equivalent to $\texttt{PR(p)} \rightarrow \neg\texttt{OP(p)} \wedge \texttt{OP(p)} \rightarrow \neg\texttt{PR(p)}$

iii. Subcontraries:
  (a) $\neg\texttt{(}\neg\texttt{NO(p)} \wedge \neg\texttt{PE(p))}$ equivalent to $\neg\texttt{NO(p)} \rightarrow \texttt{PE(p)} \wedge \neg\texttt{PE(p)} \rightarrow \texttt{NO(p)}$
  (b) $\neg\texttt{(}\neg\texttt{NO(p)} \wedge \neg\texttt{OM(p))}$ equivalent to $\neg\texttt{NO(p)} \rightarrow \texttt{OM(p)} \wedge \neg\texttt{OM(p)} \rightarrow \texttt{NO(p)}$
  (c) $\neg\texttt{(}\neg\texttt{PE(p)} \wedge \neg\texttt{OM(p))}$ equivalent to $\neg\texttt{PE(p)} \rightarrow \texttt{OM(p)} \wedge \neg\texttt{OM(p)} \rightarrow \texttt{PE(p)}$

iv. Contradictories:
  (a) $\neg\texttt{(OB(p)} \wedge \texttt{OM(p))} \wedge \neg\texttt{(}\neg\texttt{OB(p)} \wedge \neg\texttt{OM(p))}$ equivalent to $\texttt{OB(p)} \leftrightarrow \neg\texttt{OM(p)}$
  (b) $\neg\texttt{(PR(p)} \wedge \texttt{PE(p))} \wedge \neg\texttt{(}\neg\texttt{PE(p)} \wedge \neg\texttt{PE(p))}$ equivalent to $\texttt{PE(p)} \leftrightarrow \neg\texttt{PR(p)}$
  (c) $\neg\texttt{(OP(p)} \wedge \texttt{NO(p))} \wedge \neg\texttt{(}\neg\texttt{OP(p)} \wedge \neg\texttt{NO(p))}$ equivalent to $\texttt{OP(p)} \leftrightarrow \neg\texttt{NO(p)}$

In order to minimize the set of symbols and avoid redundancies between (21) and (22), this paper adopts only the deontic modalities OB, PE, and OP (obligatory, permitted, and optional). By virtue of the bi-implications in (22.iv.(a)–(c)), the remaining modalities, i.e., OM, PR, and NO (omissible, prohibited, and non-optional), are represented as ¬OB, ¬PE, and ¬OP, respectively. Nonetheless, the narrative that follows will frequently use the terms “prohibitions” and “prohibited” for readability, even though they are formally represented as non-permissions, given their common occurrence in legislative texts.

Thanks to the bi-implications in (22.iv.(a)–(c)), all entailments in the Deontic Traditional Scheme can be expressed solely in terms of OB, PE, and OP, along with the Boolean connectives ¬, ∧, →, and ↔. The complete set of entailments is thus reduced to only those presented in (23), as all others follow from these three.

(23) a. $\texttt{PE(p)} \leftrightarrow \neg\texttt{OB(}\neg\texttt{p)}$

b. $\texttt{OP(p)} \leftrightarrow (\neg\texttt{OB(p)} \wedge \neg\texttt{OB(}\neg\texttt{p))}$

c. $\texttt{OB(p)} \rightarrow \texttt{PE(p)}$

Although, as explained earlier, it is widely accepted that the entailments of the Deontic Traditional Scheme accurately reflect our intuitions about the six deontic modalities, most deontic logics proposed in the literature formalize only the so-called “main” deontic modality, namely obligatoriness (OB).

In particular, according to (Goble, 2013), the benchmark deontic logic, known as "Standard Deontic Logic", is the propositional modal logic that axiomatizes $\mathtt{OB}$ as shown in (24). In (24), $\mathtt{D}$, $\mathtt{C}$, $\mathtt{NM}$, $\mathtt{P}$, and $\mathtt{N}$ denote the axioms' names, while $\Box$ and $\Diamond$ represent the classical necessity and possibility operators. The axioms listed in (24) assume the standard deductive principles of normal modal logic, e.g., the axiom K for $\Box$, and form the basis of the normal modal logic $\mathtt{KD}$[14] for the operator $\mathtt{OB}$. Furthermore, in all formulae presented in this section, the substitution principle is assumed to hold. Therefore, symbols such as $\mathtt{p}$, $\mathtt{q}$, etc., should be understood not as propositional atomic formulas but as metavariables that can be uniformly replaced by arbitrary non-deontic formulas.

(24)
- (D) $\mathtt{OB(p)} \rightarrow \neg\mathtt{OB(\neg p)}$
- (C) $(\mathtt{OB(p)} \wedge \mathtt{OB(q)}) \rightarrow \mathtt{OB(p \wedge q)}$
- (NM) $\Box(\mathtt{p}\rightarrow\mathtt{q}) \rightarrow (\mathtt{OB(p)} \rightarrow \mathtt{OB(q)})$
- (P) $\mathtt{OB(p)} \rightarrow \Diamond(\mathtt{p})$
- (N) $\Box(\mathtt{p}) \rightarrow \mathtt{OB(p)}$

It is straightforward to see that the axiomatization in (24) is not suitable for the objectives of this paper, particularly due to the axiom $\mathtt{D}$. This axiom states that the obligation of a proposition $\mathtt{p}$ contradicts the obligation of its negation: the two deontic statements cannot both hold simultaneously. Thus, referring to the first example seen above in (1), where $\mathtt{OB(l)}$ corresponds to "It is obligatory to leave the building" and $\mathtt{OB(\neg l)}$ corresponds to "It is obligatory not to leave the building", axiom ($\mathtt{D}$) entails that $\mathtt{OB(l)} \wedge \mathtt{OB(\neg l)} \rightarrow \bot$.

By contrast, this paper aims to represent $\mathtt{OB(l)}$ and $\mathtt{OB(\neg l)}$ as *conflicting*, rather than *contradictory*, deontic statements. Therefore, to meet our objectives, a different axiomatization of the deontic operators must be adopted. The first step in this direction is to analyze conflict-tolerant deontic logics proposed in the literature, with (Goble, 2013) providing, in our view, the most comprehensive survey.

4.1 Conflict-tolerant deontic logics proposed in the literature

Since the axiomatization in (24) treats conflicts as contradictions, (Goble, 2013) identifies the first Desideratum for a conflict-tolerant deontic logic as follows:

(25) **Desideratum #1**: A conflict-tolerant deontic logic regards conflicts between pairs of deontic statements as *consistent*. In (Goble, 2013), a conflict is intuitively defined as a situation in which "an agent ought to do a number of things, each of which is possible for the agent, but it is impossible for the agent to do them all". Note that this definition differs from the one proposed by Hans Kelsen and discussed in Section 2 above; this difference underpins the key insights of the research presented here, as will be explained below.

As exemplified above, what specifically prevents to achieve Desideratum #1 with the axiomatization in (24) is the axiom $\mathtt{D}$. However, simply removing $\mathtt{D}$ is also unsatisfactory,

[14] `https://plato.stanford.edu/entries/logic-modal`

because the remaining axioms would lead to the so-called “deontic explosion”: once a conflict between two or more deontic statements is inferred, it would imply that everything is obligatory, which is clearly meaningless. In light of this, (Goble, 2013) proposes a second Desideratum for a conflict-tolerant deontic logic:

(26) **Desideratum #2**: A conflict-tolerant deontic logic must not produce deontic explosion from conflicting deontic statements. In other words, if two deontic statements conflict, the logic must *not* infer that everything is obligatory.

Finally, (Goble, 2013) introduces a third Desideratum, which generally states that conflict-tolerant deontic logics must be able to capture our intuitions regarding obligatoriness and related concepts:

(27) **Desideratum #3**: A conflict-tolerant deontic logic should provide a plausible explanation for the apparent validity of several paradigm arguments.

However, although (27) refers to paradigm arguments *in general*, (Goble, 2013) focuses primarily on one particular argument known in the literature as “The Smith argument”, along with several variants called the “Jones”, the “Roberts”, and the “Thomas” arguments. The Smith argument, originally introduced in (Horty, 1994), states the following:

(28) *From*: Smith ought to fight in the army or perform alternative national service.

And: Smith ought not to fight in the army.

It is intuitive to conclude that: Smith ought to perform alternative national service.

The axioms listed in (24) cannot derive the Smith argument, as none of them involve the Boolean connective $\vee$. Additional axioms must therefore be introduced to account for it. The Jones, Roberts, and Thomas arguments are more elaborate variants of the Smith argument, in which disjunction further interacts with the Boolean connective $\wedge$.

On the other hand, we observe that Desideratum #3 refers to paradigm arguments *in general*, even though (Goble, 2013) focuses primarily on the Smith argument and its named variants. While (Goble, 2013) states that these cases *include* the relevant paradigms, no additional paradigm scenarios are examined. In contrast, this paper considers a broader range of cases involving conflicts among deontic statements, in particular those derived from Kelsen’s categorization of conflicts and illustrated in Section 2 above. The conflict-tolerant deontic logics reviewed in (Goble, 2013) appear to be inadequate to address these cases, as will be argued later in the paper.

The following three subsubsections briefly summarize the main approaches that have been proposed in the literature to address irresolvable conflicts. These approaches are categorized in (Goble, 2013) as: (1) Revisionist strategies, (2) Paraconsistent deontic logics, and (3) Other radical strategies.

4.1.1 Revisionist strategies

Revisionist strategies are approaches that represent and reason about conflicts in deontic logic using axiomatizations alternative to the one in (24). Although (Goble, 2013) introduces twelve different deontic logics of this kind, only one, called `BDL`, along with a slight variant called `BDLcc`, satisfies all three Desiderata.

BDL retains only the axiom (C) from (24), while it rejects (D), (P), and (N). The axiom (NM) is partially rejected in the sense that it is replaced by a weaker axiom, (RBE), whose inferential strength is significantly more limited than that of (NM). In addition, two further axioms, (DDS) and (M), are introduced to support derivations involving the Boolean connectives $\vee$ and $\wedge$. These derivations are necessary to explain the Smith argument and its variants. The full axiomatization of BDL is presented in (29).

In (29), the symbol "$\leftrightarrow_{\mathtt{A}}$" denotes a restricted version of the standard bi-implication operator $\leftrightarrow$. This restricted bi-implication is defined in such a way as to avoid deontic explosion (see (Goble, 2013) for formal details).

(29) (M) $\mathtt{OB}(\mathtt{p}\wedge\mathtt{q}) \rightarrow \mathtt{OB}(\mathtt{p})$

(C) $(\mathtt{OB}(\mathtt{p}) \wedge \mathtt{OB}(\mathtt{q})) \rightarrow \mathtt{OB}(\mathtt{p}\wedge\mathtt{q})$

(DDS) $(\mathtt{OB}(\mathtt{p}\vee\mathtt{q}) \wedge \mathtt{OB}(\neg\mathtt{q})) \rightarrow \mathtt{OB}(\mathtt{p})$

(RBE) $(\mathtt{p} \leftrightarrow_{\mathtt{A}} \mathtt{q}) \rightarrow (\mathtt{OB}(\mathtt{p}) \leftrightarrow \mathtt{OB}(\mathtt{q}))$

The present paper considers the axiomatization of BDL to be correct, with one minor exception concerning the axiom (RBE), which will be discussed below.

In particular, the authors of this paper fully support the decision to reject axioms (D), (P), and (N) from the axiomatization of a conflict-tolerant deontic logic.

Axiom (D) must be rejected because, as explained earlier, it infers that conflicts are contradictions in cases where both a proposition p and its negation are obligatory.

Axiom (P), on the other hand, appears counter-intuitive. It states that if something is obligatory, then it is possible. However, it is easy to imagine situations where this does not seem to hold. For example, (4) above illustrates a case in which paying in cash is prohibited, even though it is *impossible* to avoid doing so, since one must pay at a parking meter in Sketty that only accepts cash.

Similar considerations apply to axiom (N), which asserts that if something is necessary, then it is obligatory. This implication also conflicts with our intuitions. In the scenario just considered, for instance, it is necessary to pay in cash at the parking meter in Sketty, but this is *not* obligatory; in fact, doing so is prohibited.

In this paper, we adopt only the axioms (M), (C), and (DDS) from BDL. Axiom (RBE), by contrast, is not relevant to the examples discussed here and is therefore omitted. Furthermore, normative reasoning involving (bi-)implications among obligations relies on concepts that fall outside the scope of this paper. These concepts have been recently explored in (Robaldo and Liga, 2025), where the axiom (RBE) is implemented in the computational ontology presented there, using the standard bi-implication operator $\leftrightarrow$ instead of $\leftrightarrow_{\mathtt{A}}$. The rationale is that, as will become clear below, deontic explosion is avoided in our framework by adopting the same criterion used in the paraconsistent deontic logics discussed in the next subsection. As a result, it is unnecessary to (also) restrict the bi-implication operator, as is done in BDL. For further details, we refer the reader to (Robaldo and Liga, 2025), as we do not repeat its formalization here.

4.1.2 Paraconsistent deontic logics

Revisionist strategies propose axiomatizations, within classical modal logic enriched with the operator OB, that represent conflicts as consistent formulae while avoiding deontic explosion. In particular, to prevent such explosion, either the axiom (C) or the axiom (NM) is suitably restricted to block the inferences that lead to it. For instance, the logic BDL replaces (NM) with its restricted version (RBE) to achieve this goal.

Paraconsistent deontic logics, by contrast, avoid deontic explosion by rejecting a specific inference rule of classical modal logic: Ex falso quodlibet, which, together with (C) and (NM), is responsible for deontic explosion. By blocking Ex falso quodlibet, the axiomatization of the OB operator can retain the full inferential strength of (C) and (NM).

Ex falso quodlibet states that any proposition can be derived from a contradiction, i.e., $\bot \rightarrow \mathtt{P}$, for any P. However, Ex falso quodlibet itself is derived from two well-known inference rules associated with the Boolean connective $\vee$ in standard logic: Disjunction Introduction, which states that $\mathtt{A} \rightarrow (\mathtt{A} \vee \mathtt{B})$, and Disjunctive Syllogism, which states that $((\mathtt{A} \vee \mathtt{B}) \wedge \neg \mathtt{B}) \rightarrow \mathtt{A}$. Therefore, in order to block Ex falso quodlibet, at least one of these two inference rules must be rejected.

According to (Goble, 2013), the trend in the literature on paraconsistent and relevant logics is to block Disjunctive Syllogism. Nevertheless, without Disjunctive Syllogism, deriving the Smith argument, and thereby satisfying Desideratum #3, becomes difficult. (Goble, 2013) discusses some approaches to accommodate the Smith argument in paraconsistent deontic logics that lack Disjunctive Syllogism.

On the other hand, regarding relevant logics more specifically, it is important to note that “the rejection of Disjunctive Syllogism, however, has become one of the most controversial aspects of relevance logic.”[15]. Even though most relevant logics opt to reject Disjunctive Syllogism, the rule is at least admissible in some relevant logics, since it is theorem-preserving: if $\mathtt{A} \vee \mathtt{B}$ and $\neg \mathtt{A}$ are theorems, so is B (Jago, 2020). Moreover, weaker relevant logics explore the consequences of including Disjunctive Syllogism as a primitive rule (Robles and Méndez, 2010), (Robles and Méndez, 2011), with semantics grounded in Routley-Meyer ternary frames (Routley and Meyer, 1973), (Routley and Meyer, 1972a), (Routley and Meyer, 1972b), under which these logics remain sound and complete.

Another crucial aspect to take into account is the strong connection between Disjunctive Syllogism and the cut rule, a well-established property that plays a central role in the design of proof systems and reasoning tools for a given logic (Gentzen, 1964). Removing Disjunctive Syllogism may therefore compromise the ability to develop sound and complete provers for relevant logics. Notably, the cut rule forms the basis of the Prolog programming language, arguably the most widely used logic programming language. This connection supports a preference for Disjunctive Syllogism over Disjunction Introduction also from a practical standpoint.

It is particularly these practical considerations that led us to adopt Disjunctive Syllogism and reject Disjunction Introduction in the proposed computational ontology. As stated in the Introduction, our aim is to develop a framework that, like Prolog, can be concretely executed. For this reason, we tend to prioritize practical applicability. However, it is also evident that the choice between Disjunctive Syllogism and Disjunction Introduction is not trivial and cannot be resolved lightly. It thus remains an open problem. Future work may explore and integrate into the proposed computational ontology non-standard versions of these two rules that avoid generating Ex falso quodlibet.

[15]Citation from `https://plato.stanford.edu/entries/logic-relevance`

4.1.3 Other radical strategies

In (Goble, 2013), paraconsistent deontic logics are classified as a type of "radical strategy" for addressing conflicts among obligations. However, in this paper, we prefer to distinguish them from other radical strategies, as they share with revisionist strategies the feature of being grounded in classical modal logic, on top of which the operator $\mathtt{OB}$ is defined and axiomatized.

By contrast, the other radical strategies presented in (Goble, 2013) differ from both revisionist strategies and paraconsistent logics in that they introduce an *additional upper level* that governs and constrains inferences at the base level.

Three examples of such radical strategies, discussed in (Goble, 2013), include Two-phase deontic logic (van der Torre and Tan, 2000), imperatival approaches (e.g., (Hansen, 2008) and (Horty, 2012)), and Adaptive Deontic Logics (e.g., (Goble, 2014) and (van De Putte, Beirlaen, and Meheus, 2019)).

In Two-phase deontic logic, the upper level controls and constrains the order in which inference rules are applied to $\mathtt{OB}$ and to the classical modal logic operators. Specifically, this logic stipulates that aggregation rules, such as axiom ($\mathtt{C}$), must always be applied *before* distribution rules, such as axiom ($\mathtt{NM}$). (van der Torre and Tan, 2000) demonstrate that imposing this ordering allows the logic to represent conflicts as consistent formulae, to correctly derive the Smith argument, and to avoid deontic explosion. Nevertheless, while this solution works technically for the examples considered, (Goble, 2013) questions the underlying rationale. Why should the rules be applied in that particular order? What intuitions about the nature of conflict does this order reflect? (van der Torre and Tan, 2000) do not seem to offer answers to these questions.

Imperatival approaches assume that obligations are always grounded in commands or directives. The core intuition is that agents are obliged to act in a certain way because they have been *ordered* to do so, by law, or by an authority empowered to issue such commands. In this approach, the upper level comprises a set of commands associated with the obligations in the lower level. Conflicts among obligations are interpreted as conflicts among the associated commands. Agents are assumed to comply with a maximal set of non-conflicting obligations, and, unless otherwise specified (e.g., through prioritization), they are free to choose which obligations to include in that set. Once such a set is identified, since it contains no conflicts, there is no need to restrict the underlying classical modal logic or $\mathtt{OB}$'s axiomatization. Note that imperatival approaches are inherently non-monotonic, since agents may have multiple valid choices when resolving conflicting orders.

Finally, Adaptive Deontic Logics are dynamic and non-monotonic logical systems in which the application of inference rules may change over time. In their general form, Adaptive Logics are defined as triples $\langle\mathtt{LLL}, \Omega, \mathtt{Strategy}\rangle$, where $\mathtt{LLL}$ is a basic logic with specified properties (e.g., reflexivity and monotonicity, see (Goble, 2014) for details), Ω is a set of *abnormalities* (i.e., problematic formulae the logic must "adapt" to), and $\mathtt{Strategy}$ is a set of rules that determine *how* the inference process adjusts when abnormalities are detected. Under this schema, an Adaptive Deontic Logic may be constructed by taking $\mathtt{LLL}$ as classical modal logic enriched with the operator $\mathtt{OB}$, Ω as the set of all formulae denoting conflicts, i.e., $\Omega \equiv A : \exists C\,[A = \mathtt{OB}(C) \wedge \mathtt{OB}(\neg C)]$, and $\mathtt{Strategy}$ as a set of rules that block undesirable consequences (chiefly, deontic explosion) as soon as a conflict or abnormality is derived. Multiple configurations of inference rules within

LLL and Strategy can be devised to account for the Smith argument and, more generally, to satisfy Desideratum #3, while avoiding unwanted inferences.

4.2 Extending Hobbs's framework with deontic modalities

In the final subsection of this section, Hobbs's framework from Section 3 is extended to incorporate deontic modalities, the Deontic Traditional Scheme (i.e., the axioms in (23)), and the axioms (M), (C), and (DDS) from BDL, needed to achieve Desideratum #3.

Desiderata #1 and #2 will instead be achieved by rejecting certain inference rules from classical logic, following the approach of paraconsistent deontic logics. In particular, Section 6 will introduce rules that derive a conflict, rather than a contradiction, in all the cases identified by Kelsen and discussed above in Section 2. Conversely, as already noted, the proposed computational ontology will reject the rule of Disjunctive Introduction, thereby preventing Ex falso quodlibet from occurring.

Deontic modalities can be implemented in Hobbs's framework by introducing three unary predicates: Obligatory, Permitted, and Optional. Similar to Rexist, these predicates express modalities, indicating that the eventuality in their argument is obligatory, permitted, or optional, respectively. In line with what has been explained earlier in this section, omissions, prohibitions, and non-optionality are represented as ¬Obligatory, ¬Permitted, and ¬Optional, respectively.

The proposed computational ontology establishes that, like the Subsist modality, the three deontic modalities apply exclusively to abstract eventualities. In other words, deontic modalities do not pertain to specific eventualities that actually occur in the state of affairs; rather, they concern eventualities that exist only in abstraction and, in turn, refer to *sets* of concrete eventualities that do occur: namely, all those that fulfill an obligation or are permitted under a given permission. See Subsection 5.5 below for the corresponding implementation in SPARQL*.

The axioms in (23) corresponds to the following bi-implications in Hobbs:

(30)
a. $\forall_{\mathtt{e,en}}[\mathtt{not(en,\ e)} \rightarrow (\mathtt{Permitted(e)} \leftrightarrow \neg\mathtt{Obligatory(en)})]$

b. $\forall_{\mathtt{e,en}}[\mathtt{not(en,\ e)} \rightarrow (\mathtt{Optional(e)} \leftrightarrow (\neg\mathtt{Obligatory(e)} \wedge \neg\mathtt{Obligatory(en)}))]$

c. $\forall_{\mathtt{e}}[\mathtt{Obligatory(e)} \rightarrow \mathtt{Permitted(e)}]$

There is a key conceptual difference between the formulae in (30) and those in (23): while the latter use a single type of negation (¬), the former employ *two* types of negation: ¬ and not. The Deontic Traditional Scheme considers only whole propositions and applies negation in the same way to both the proposition and the outer deontic modality operators. In contrast, the approach proposed here focuses on *eventualities*, e.g., actions, rather than propositions. Given an eventuality e, the predicate not is used to refer to its *opposite* eventualities, i.e., those that do not really exist when e does and vice versa.

The operator ¬ is still used to negate deontic modalities, as in the Deontic Traditional Scheme. However, there is another key difference with respect to the Deontic Traditional Scheme: as mentioned earlier, whenever a deontic modality both holds and does not hold for the same eventuality, a conflict, rather than a contradiction, is inferred. In other words, the following rule, which defines the standard semantics for the operator ¬:

(31) $\forall_{\texttt{e}}[\texttt{(P(e)}\wedge\neg\texttt{P(e))}\rightarrow\bot]$

does not hold when `P` is one of `Obligatory`, `Permitted`, or `Optional`. This is one of the inference rules from classical logic that the proposed computational ontology rejects.

The axioms (`M`), (`C`), and (`DDS`) from `BDL` are implemented as follows:

(32)

a. $\forall_{\texttt{ea, e1, e2}}[\texttt{and(ea, e1, e2)}\rightarrow$
$\quad\quad(\texttt{Obligatory(ea)}\leftrightarrow(\texttt{Obligatory(e1)}\wedge\texttt{Obligatory(e2)}))]$

b. $\forall_{\texttt{eo, e1, e2}}[((\texttt{or(eo, e1, e2)}\vee\texttt{or(eo, e2, e1)})\wedge\texttt{not(en2, e2)})\rightarrow$
$\quad\quad((\texttt{Obligatory(eo)}\wedge\texttt{Obligatory(en2)})\rightarrow\texttt{Obligatory(e1)})]$

In particular, `(M)` and `(C)` correspond to (32.a) while `(DDS)` corresponds to (32.b).

With the new introduce deontic modalities, it is now possible to represent, for instance, that John is obliged to not leave and that John is prohibited to pay in cash. These are respectively represented as in (33.a) and (33.b).

(33)

a. $\texttt{Obliged(enlj)}\wedge\texttt{not(enlj, elj)}\wedge\texttt{Leave(elj)}\wedge\texttt{has-agent(elj, John)}$

b. $\neg\texttt{Permitted(epjc)}\wedge\texttt{Pay(epjc)}\wedge\texttt{has-agent(epjc, John)}\wedge$
$\quad\quad\texttt{has-instrument(epjc, cash)}$

It must be pointed out, however, that, unlike facts, represented in Hobbs's framework via the `Rexist` or `Subsist` modalities, what we typically need to represent through the deontic modalities `Obligatory`, `Permitted`, and `Optional` does not usually concern a single individual (e.g., John, in the examples in (33)), but rather a *set* of individuals.

This is especially true when formalizing norms from legislation: norms rarely apply to a single individual. Instead, they generally specify that all individuals belonging to certain categories or meeting specific criteria are obliged, prohibited, or permitted to perform certain actions. These individuals are typically referred to as the norm's *bearers*.

For this reason, standard legal theory usually represents norms as *if-then rules*, where the antecedent specifies the conditions an entity must satisfy to qualify as the norm's bearer, and the consequent specifies what the bearer is obliged, prohibited, or permitted to do (see (Searle, 1995), among others).

Consequently, in the proposed formalization, every eventuality associated with one of the deontic modalities must always include the `has-agent` role: this role identifies the bearer specified in the norm's antecedent.

This assumption is not new in the literature on deontic logic either. In fact, there is extensive research on the interaction between deontic modalities and agency[16], with a recent approach presented in (Frijters, Meheus, and van De Putte, 2021). This body of work has led to the development of the so-called "normative positions", which formalize the well-known Hohfeldian legal relations (Sergot, 2013).

Sometimes the bearers of norms are *implicit*, although they can be easily deduced from the context of the norms, even in the case of hypothetical norms such as the very first ones considered in this paper, which are repeated in (34) for the reader's convenience:

[16]See `https://plato.stanford.edu/entries/logic-deontic/#DeonLogiAgen`

(34) a. It is obligatory to leave the building.

b. It is obligatory to not leave the building.

Naturally, the bearer of (34.a-b) is "every human inside the building", even though this is not explicitly stated. In other words, the two obligations do not apply to cats, dogs, or other inanimate entities inside the building, as they are incapable of understanding the norms. Nor do they apply to humans *outside* the building. In general, legal norms apply to entities with decisional capacity. These typically include humans, but also *legal* persons such as companies or, in some cases, robots.

Under these assumptions, (34.a-b) are represented in Hobbs's framework by the if-then implications in (35.a-b), respectively:

(35) a.
$$\forall_{u}[\exists_{e1}[\texttt{Rexist(e1)}\wedge\texttt{Be(e1)}\wedge\texttt{has-agent(e1, u)}\wedge \texttt{Human(u)}\wedge\texttt{has-inside-location(e1, B)}]\rightarrow \exists_{e2}[\texttt{Obligatory(e2)}\wedge\texttt{Leave(e2)}\wedge\texttt{has-agent(e2, u)}\wedge \texttt{has-from-location(e2, B)}]]$$

b.
$$\forall_{u}[\exists_{e1}[\texttt{Rexist(e1)}\wedge\texttt{Be(e1)}\wedge\texttt{has-agent(e1, u)}\wedge \texttt{Human(u)}\wedge\texttt{has-inside-location(e1, B)}]\rightarrow \exists_{en2}[\texttt{Obligatory(en2)}\wedge\texttt{not(en2, e2)}\wedge\texttt{Leave(e2)}\wedge\texttt{has-agent(e2, u)}\wedge \texttt{has-from-location(e2, B)}]]$$

Another example is shown in (36); this if-then rule formalizes the sentence in (4.a) above, repeated in (36) for the reader's convenience:

(36) Whoever parks in a parking spot is obliged to pay £3 at the parking meter associated with that spot.

$$\forall_{a,pm}[\exists_{e1,p}[\texttt{Rexist(e1)}\wedge\texttt{Park(e1)}\wedge\texttt{has-agent(e1, a)}\wedge\texttt{has-location(e1, p)}\wedge \texttt{parkingSpot(p)}\wedge\texttt{associated-with(p, pm)}]\rightarrow \exists_{e2}[\texttt{Obligatory(e2)}\wedge\texttt{Pay(e2)}\wedge\texttt{has-agent(e2, a)}\wedge \texttt{has-object(e2, £3)}\wedge\texttt{has-recipient(e2, pm)}]$$

The formula in (36) states that if John or anyone else parks in a parking spot, (36) will infer that the individual, i.e., the agent performing the parking action, is obliged to pay £3 at the parking meter associated with that spot.

5 Implementing Hobbs's extended framework in RDF* and SPARQL*

This paper aims to define a conflict-tolerant deontic logic compatible with RDF, the standard used to encode data on the Semantic Web.

As is well known, RDF is a graph-based data model that merely *represents* knowledge, yet it does not include inference mechanisms to derive logical consequences from explicitly asserted information. Consequently, RDF alone is insufficient for developing applications for compliance checking or other AI systems based on symbolic reasoning, as it lacks native reasoning capabilities. To enable reasoning, RDF must be used in conjunction with a dedicated reasoning language, such as OWL, SWRL, LegalRuleML, or the other languages introduced in the Introduction and used in prior literature.

This paper, which focuses exclusively on irresolvable conflicts, employs knowledge bases and inference rules encoded in RDF* and SPARQL*. These are extensions of standard RDF and SPARQL designed to support and simplify reification, a core feature of Hobbs's formalization and, therefore, also of the framework proposed here.

As is well known, the standard RDF reification vocabulary is verbose and impractical for large-scale applications. To address this limitation, RDF* has been proposed as an extension of RDF to streamline the reification process[17]. RDF* enables a more concise and efficient representation of metadata about triples, significantly reducing the verbosity of traditional RDF reification. It is important to note, however, that RDF* does not extend the expressivity or semantics of standard RDF; rather, it only introduces a more compact syntax that allows for statements about statements to be embedded directly within the triple structure. This makes RDF* particularly useful for practical implementations that require reasoning over embedded statements. RDF* is already supported by numerous RDF processing frameworks, including Apache Jena[18], which is used in the GitHub repository associated with this paper.

Therefore, the core semantics of the proposed formalization remain those of RDF[19] and SPARQL[20] (Perez, Arenas, and Gutierrez, 2006). The automated reasoner operates as a SPARQL rule engine, iteratively executing rules in the form `CONSTRUCT-WHERE` until no additional triples can be derived. This approach mirrors the behavior of standard OWL reasoners such as HermiT (Glimm et al., 2014), which similarly reapply OWL axioms until no new RDF triples are inferred. All other facets of Hobbs's logical framework and the Deontic Traditional Scheme are "embedded" within the semantics of RDF and SPARQL, as detailed in the following subsections.

While this logical core establishes the necessary semantic foundations, it is not, by itself, sufficient for a full-fledged compliance-checking framework. Real-world use cases also involve *solvable* conflicts, in which norms defeasibly overrides others, and they demand reasoning engines that scale efficiently to large RDF datasets (Robaldo et al., 2023). Defeasibility can be captured by replacing SPARQL rules with SHACL-SPARQL rules as in (Anim, Robaldo, and Wyner, 2024). SHACL, as noted in the Introduction, permits rule prioritization and thus rule overriding. To address efficiency at scale, future work will investigate the use of ASP-based reasoners such as DLV2 (Calimeri et al., 2020) and Vadalog (Bellomarini et al., 2022), which are fully compatible with RDF and SPARQL.

The following subsections explain how the formalizations presented in the previous two sections have been implemented using RDF* and SPARQL*.

[17]`https://www.w3.org/2022/08/rdf-star-wg-charter`

[18]`https://jena.apache.org/`

[19]`https://www.w3.org/TR/rdf11-mt`

[20]`https://www.w3.org/TR/sparql11-query`

5.1 Basic classes, properties, and inference rules

All predicates introduced earlier in the paper can be directly encoded in RDF. For example, the formula shown above in (7.a) can be directly implemented in RDF by mapping the predicates `Leave` and `Rexist` to homonymous RDF classes, `has-agent` to a homonymous RDF property, and `elj` and `John` to homonymous RDF individuals.

Furthermore, the class `Leave` is asserted as an individual of an additional class `EventualityType`, the class `Rexist` as an individual of an additional class `Modality`, and the property `has-agent` as an individual of an additional class `ThematicRole`. Formula (7.a) is then encoded in RDF, Turtle syntax[21], as shown in (37). Formula (7.b) can be similarly represented by introducing additional RDF classes and properties to encode the predicates `Pay`, `has-object`, and `has-instrument`.

(37)
```
:Eventuality a rdfs:Class. :ThematicRole a rdfs:Class.
:Modality a rdfs:Class. :EventualityType a rdfs:Class.
:Rexist a rdfs:Class,:Modality.
soa:Leave a rdfs:Class,:EventualityType.
soa:has-agent a rdf:Property,:ThematicRole.
soa:elj a :Rexist,soa:Leave; soa:has-agent soa:John.
```

The empty prefix "`:`" is associated with the following namespace:

`https://w3id.org/ontology/conflict-tolerantdeontictraditionalscheme#`

This namespace is used for RDF resources defined within the proposed computational ontology. We also introduce the prefix "`soa:`" for RDF resources that represent states of affairs, i.e., the ABoxes encoding the examples:

`https://w3id.org/ontology/conflict-tolerantdeontictraditionalscheme#soa`

The SPARQL rules, written in the form `CONSTRUCT-WHERE`, are also part of the proposed computational ontology. Specifically, the ontology includes a dedicated class `InferenceRule`, whose individuals represent the SPARQL rules. A special RDF property, `has-sparql-code`, links each of these individuals to the string encoding of the corresponding rule in standard SPARQL v1.1.

In all examples shown below and in the GitHub repository, the inference rules are represented as anonymous RDF individuals, which conform to the following template:

(38)
```
[a :InferenceRule;
  :has-sparql-code """CONSTRUCT{...}WHERE{...}"""]
```

Two examples of inference rules are those implementing the pragmatic implicatures shown above in (14) and (16). (14) states that if the state of affairs contains an abstract eventuality `ea`, and another eventuality `ei` of the same type as `ea`, such that the thematic roles of `ei` form a superset of those of `ea` and share identical values for the overlapping roles, then it is (pragmatically and contextually) assumed that `ei` instantiates `ea`. This is implemented in SPARQL as shown in (39). Note that the "$\neg\exists$" conditions in (14) are directly implemented in corresponding SPARQL `NOT EXISTS` clauses.

[21] `https://www.w3.org/TR/turtle`

```
(39) [a :InferenceRule; :has-sparql-code """
      CONSTRUCT{?ei :instantiates ?ea}
      WHERE{?ea a :AbstractEventuality,?ET. ?ei a ?ET. ?ET a :EventualityType.
              NOT EXISTS{?tr a :ThematicRole. ?ea ?tr ?va.
                                                 NOT EXISTS{?ei ?tr ?vi}}
              NOT EXISTS{?tr a :ThematicRole. ?ea ?tr ?va. ?ei ?tr ?vi.
                                                   FILTER(?va!=?vi)}}"""]
```

(16) states that if two eventualities `ei1` and `ei2` share the same type, modality, and agent, then the state of affairs includes an additional abstract eventuality that is instantiated by both eventualities; this is implemented as follows:

```
(40) [a :InferenceRule; :has-sparql-code """
      CONSTRUCT{?ei1 :instantiates ?ea. ?ei2 :instantiates ?ea}
      WHERE{?ei1 a ?ET,?M; soa:has-agent ?a. ?ei2 a ?ET,?M; soa:has-agent ?a.
          ?ET a :EventualityType. ?M rdf:type/rdfs:subClassOf* :Modality.
          FILTER(STR(?ei1)<STR(?ei2))  BIND(BNode() AS ?ea)
          NOT EXISTS{?ei1 :instantiates ?ear. ?ei2 :instantiates ?ear}}"""]
```

Note the `FILTER` condition in the rule in (40): the rule is triggered only once for each pair `?ei1` and `?ei2`. If `?ei1` is lexicographically smaller than `?ei2`, the rule is triggered for the *ordered* pair (`?ei1`, `?ei2`), but not for the reverse ordered pair (`?ei2`, `?ei1`), and vice versa. This ensures that only one abstract eventuality `?ea` is created via the SPARQL command `BNode`, rather than two.

In connection with this, the rule in (40) also includes a `NOT EXISTS` clause to prevent infinite execution loops. Since the automated reasoner re-applies the rules until no new triples is inferred, omitting this clause would result in repeated creation of new anonymous nodes and two additional triples linking `?ei1` and `?ei2` to those nodes via the property `instantiates`. In contrast, with this clause, the rule triggers only once, specifically only if `?ei1` and `?ei2` do not already instantiate the same abstract eventuality.

5.2 Axiomatizing $\neg$, $\wedge$, and $\vee$ in RDF* and SPARQL*, and deriving contradictions: semantical embedding

As explained earlier, the predicates `not`, `and`, and `or` of Hobbs's framework do *not* correspond directly to the boolean connectives $\neg$, $\wedge$, and $\vee$. Instead, these connectives are used to define the *semantics* of `not`, `and`, and `or` in relation to the different modalities. This is achieved by introducing axioms that constrain the modalities of the eventualities connected by `not`, `and`, and `or` based on the modalities assigned to the others. These axioms make use of the boolean connectives $\neg$, $\wedge$, and $\vee$.

Nevertheless, these axioms cannot be directly implemented in RDF and SPARQL. This is because RDF does not natively support operators for negation and disjunction, i.e., its vocabulary does not include constructs corresponding to $\neg$ and $\vee$. By contrast, one could argue that RDF implicitly supports $\wedge$, since RDF triples are jointly true; in other words, the knowledge graph can be interpreted as the logical conjunction of all its triples. Although this interpretation is valid, it should be noted that conjunctions might be embedded within negations or disjunctions, e.g., in the formula $\texttt{A}\lor\texttt{(B}\land\texttt{C)}\lor\neg\texttt{(D}\land\texttt{E)}$. This makes it necessary to introduce an explicit construct for representing $\land$ as well, alongside those for representing $\neg$ and $\lor$.

We also remind that RDF semantics follows the Open World Assumption, meaning that in cases where some triples do not occur in the knowledge graph, it does not follow that these triples are *false*. Instead, it is simply *unknown* whether these are true or false.

To address this representational gap, the proposed computational ontology explicitly asserts that a triple is false by stating that *its reification* holds false. This is done by asserting the reified triple as an instance of a special class called `false`. This class parallels standard negation $\neg$ in the proposed computational ontology.

RDF* provides a concise syntax to reify triples: the operator "`<< ... >>`". Thus, for example, the literal $\neg$`Rexist(e)`, which encodes that the eventuality `e` does not really exist, is represented in the proposed computational ontology as follows:

```
(41) <<soa:e rdf:type :Rexist>> rdf:type :false.
```

(41) is a standard RDF triple, where `<<soa:e rdf:type :Rexist>>` serves as the subject. This subject is an individual, i.e., an RDF resource, representing the reification of the triple "`soa:e rdf:type :Rexist`". In the technical jargon used in this paper, it explicitly refers to *the fact that* `soa:e` really exists.

The proposed computational ontology also includes the class `true`, which is the dual of `false`. When the reification of a triple is asserted as an instance of `true`, it indicates that the triple holds true in the state of affairs. However, since all triples explicitly asserted in a knowledge graph are *already* considered true, reifying and re-asserting each positive triple as an instance of `true` is redundant and could even easily lead to infinite loops. Accordingly, the proposed computational ontology does *not* include a SPARQL* rule to implement this inference.

On the other hand, as it will be illustrated below, the reverse inference can occasionally be useful: during the derivations, it may be concluded that the reification of a triple belongs to the class `true`, after which the triple itself can be safely added to the knowledge graph. This is achieved through the SPARQL* rule in (42):

```
(42) [a :InferenceRule; :has-sparql-code """
        CONSTRUCT{?s ?p ?o}WHERE{<<?s ?p ?o>> a :true}"""]
```

Having formalized truth, and more importantly, falsity, which is natively absent in RDF, it becomes possible to represent and infer *contradictions*, which naturally arise when a triple is asserted as both true and false. To this end, the SPARQL* rule in (43) is added to the proposed computational ontology. The SPARQL* rule in (43) corresponds to the rule $\forall_{\texttt{e}}\texttt{[(P(e)}\land\neg\texttt{P(e))}\rightarrow\bot\texttt{]}$, *except in cases where* `P` *denotes a deontic modality.* In such cases, as explained in the previous section, we do not aim to derive a contradiction, but rather a *conflict*. The next section will illustrate how conflicts are derived in the proposed ontology, which, as previously noted, aligns with the definitions proposed by Hans Kelsen rather than those of Lou Goble.

```
(43) [a :InferenceRule; :has-sparql-code """
       CONSTRUCT{<<<<?s ?p ?o>> a :true>> :is-in-contradiction-with
                <<<<?s ?p ?o>> a :false>>}
       WHERE{?s ?p ?o.  <<?s ?p ?o>> a :false.
             NOT EXISTS{?o a :DeonticModality}}"""]
```

In (43), the RDF property `is-in-contradiction-with` parallels the standard logical symbol $\bot$, which denotes a contradiction. However, there is a crucial distinction between the two: while `is-in-contradiction-with` is part of the vocabulary of the proposed computational ontology, the symbol $\bot$ is *not* part of the object-level logic's vocabulary. Instead, $\bot$ belongs to the meta-language used to describe the *semantics* of the logic. In standard logic, the rule $\forall_e[(P(e)\wedge\neg P(e))\rightarrow\bot]$ is therefore an *interpretative rule*, belonging to the *proof theory* rather than to the syntax of the logic. It defines how the formula $P(e)\wedge\neg P(e)$ is interpreted, mapping it to its *meaning*, denoted by the symbol $\bot$.

In other words, the proposed computational ontology *embeds* the proof theory directly into the syntax of the logic. This technique, known as "semantical embedding", is widely adopted in modern logical frameworks such as LogiKEy (Benzmüller, Parent, and van der Torre, 2020). It has also been recently applied to model deontic statements and argumentation (Pasetto and Benzmüller, 2024).

The key difference between the mentioned works and this one, however, is that our implementation is fully compatible with RDF whereas, to the best of our knowledge, all existing applications of LogiKEy have been implemented in Isabelle/HOL[22]. As stated earlier, the choice of using RDF as the underlying framework, which was set as the primary objective of our research, was made to promote interoperability, standardization, and seamless integration with existing Semantic Web technologies, while ensuring compatibility with widely adopted tools and practices in knowledge representation.

Further rules may be added to derive contradictions or other forms of meaning, e.g., conflicts, as will be discussed in the next section.

For example, as explained in Subsection 3.2, if an abstract eventuality does not subsist, then none of its partial instantiations subsist either, and none of its full instantiations really exist. Therefore, if the ontology asserts that any of them does, a contradiction can be inferred. This constraint can be enforced by the following SPARQL* rule. Note that the rule uses `instantiates+`, where the `+` symbol indicates one or more occurrences of the `instantiates` property. This is necessary because the property is *transitive.*

```
(44) [a :InferenceRule; :has-sparql-code """
       CONSTRUCT{<< <<?ea a :Subsist>> a :false>> :is-in-contradiction-with
                << <<?e a ?M>> a :true>>}
       WHERE{<<?ea a :Subsist>> a :false. ?e :instantiates+ ?ea.
             ?e a ?M. VALUES ?M {:Rexist :Subsist}}"""]
```

In addition, it is useful to introduce rules that derive a contradiction when an individual is asserted to belong to two disjoint sets. For instance, the rule in (45) infers a contradiction for any individual that is classified as both a dog and a cat. A similar rule, which enforces disjointness between the classes `AbstractEventuality` and `Instance`, is available in the GitHub repository.

[22] https://isabelle.in.tum.de/

```
(45) [a :InferenceRule; :has-sparql-code """
       CONSTRUCT{<<?i a soa:Dog>> :is-in-contradiction-with
                 <<?i a soa:Cat>>}
       WHERE{?i a soa:Dog, soa:Cat}"""]
```

Consequently, if an individual, e.g., `soa:Yoof`, is asserted to belong to both the set of dogs and the set of cats, the following triple is inferred through the rule in (45):

```
<<soa:Yoof a soa:Dog>> :is-in-contradiction-with <<soa:Yoof a soa:Cat>>
```

This triple straightforwardly reads as follows: "the fact that Yoof is a dog is in contradiction with the fact that Yoof is a cat".

Having explicitly represented truth and falsity in the proposed computational ontology, we can now introduce operators that parallel the Boolean connectives ∧ and ∨: these are the RDF properties `conjunction` and `disjunction`, respectively. Both the subjects and the objects of these properties are reifications of triples in the form "`R a tf`", where `R` is the reification of another triple and `tf` is either the class `true` or the class `false`.

The use of `conjunction` and `disjunction` is best illustrated through a few examples. The disjunction `Rexist(e1)`∨`Rexist(e2)` is represented as:

```
<<<<soa:e1 a :Rexist>> a :true>> :disjunction
<<<<soa:e2 a :Rexist>> a :true>>.
```

Note that neither `Rexist(e1)` nor `Rexist(e2)` is asserted to hold true; this is appropriate, as we know that at least one of them does hold true, but we do not know *which one*. This is precisely why the subject and object of `disjunction` are reifications: they refer to *the fact that* the two triples hold true, without actually asserting that they do.

Another example is the representation of `Rexist(e1)∨¬(Rexist(e2)∧¬Rexist(e3))`:

```
<<<<soa:e1 a :Rexist>> a :true>> :disjunction
<<<<
    <<<<soa:e2 a :Rexist>> a :true>> :conjunction
    <<<<soa:e3 a :Rexist>> a :false>>
>> a :false>>.
```

As in the previous example, it remains unknown whether `Rexist(e1)`, `Rexist(e2)`, and/or `Rexist(e3)` hold true or false.

It is now possible to introduce standard axioms for ¬, ∧, and ∨. For example, axiom (46) states that if a conjunction holds true, then each of its conjuncts also does.

```
(46) [a :InferenceRule; :has-sparql-code """
        CONSTRUCT{?s1 ?p1 ?o1.  ?s2 ?p2 ?o2.}
        WHERE{<<?s1 ?p1 ?o1>> :conjunction <<?s2 ?p2 ?o2>>}"""]
```

Conversely, in cases where a conjunction holds false, we can apply De Morgan's law `¬(P1∧P2)→(¬P1∨¬P2)` to infer a disjunction that holds true. Similarly, if a disjunction holds false, we can apply De Morgan's law `¬(P1∨P2)→(¬P1∧¬P2)` to infer a conjunction that holds true. These two inference rules can be implemented using a single SPARQL rule as in (47), where the `IF` operator allows both the truth values and the properties `:disjunction` and `:conjunction` to be swapped in the inferred triple[23]:

```
(47) [a :InferenceRule; :has-sparql-code """
        CONSTRUCT{<<?t1 a ?vn1>> ?doc <<?t2 a ?vn2>>}
        WHERE{<< <<?t1 a ?v1>> ?cod <<?t2 a ?v2>> >> a :false.
              BIND(IF(?cod=:conjunction,:disjunction,
                   IF(?cod=:disjunction,:conjunction,?doc)) AS ?doc)
              FILTER(BOUND(?doc))
              BIND(IF(?v1=:false,:true,:false) AS ?vn1)
              BIND(IF(?v2=:false,:true,:false) AS ?vn2)}"""]
```

Another axiom worth including in the proposed computational ontology, in light of the discussion from the previous section, is the Disjunctive Syllogism, implemented as follows:

```
(48) [a :InferenceRule; :has-sparql-code """
        CONSTRUCT{?t2 a ?v2}
        WHERE{{<< <<?s ?p ?o>> a ?v1>> :disjunction <<?t2 a ?v2>>}UNION
              {<<?t2 a ?v2>> :disjunction << <<?s ?p ?o>> a ?v1>>}
              FILTER((?v1=:false && EXISTS{?s ?p ?o})||
                     (?v1=:true && EXISTS{<<?s ?p ?o>> a :false}))}"""]
```

Additional axioms could similarly be added to the proposed computational ontology; however, doing so goes well beyond the scope of the present paper.

It is nonetheless important to emphasize that while many axioms of standard logic are *theoretically* valid, their direct implementation in a concrete framework such as RDF may lead to impossible or undesirable consequences. For example, the rule of Conjunctive Introduction, which states that if both `A` and `B` hold true, then their conjunction `A`$\wedge$`B` also holds true, is arguably redundant in RDF, given that, as observed above, all asserted triples in a knowledge graph are already assumed to be jointly true. A similar issue arises with Disjunction Introduction, which states that if a triple is true, then the disjunction of that triple with any other triple is also true. Generating all such disjunctions in the knowledge graph would be superfluous unless some of them are required as antecedents in other inference rules (see discussions in (Parry, 1989) and (Faroldi and van De Putte, 2023)). For this reason, restricted forms of the Disjunction Introduction rule should be adopted. For example, (Parry, 1989) argues that `A`$\rightarrow$`(A`$\vee$`B)` should be considered valid only if all propositional variables in `B` also occur in `A`. Moreover, as already discussed in Subsubsection 4.1.2, including both Disjunction Introduction and Disjunctive Syllogism in their standard forms may lead to Ex falso quodlibet. Therefore, it is essential to verify that the proposed restricted versions do not also give rise to Ex falso quodlibet.

In light of these considerations, a thorough examination of the relevant literature is necessary to determine how axioms involving $\neg$, $\wedge$, and $\vee$ from standard logic should be implemented, either within the reification-based approach adopted in the proposed computational ontology or through alternative RDF representations specifically designed for this purpose. This undertaking requires substantial further research, which, as we emphasize once again, lies beyond the scope of the present work.

[23]Note that `FILTER(BOUND(?doc))` is true only if `?doc` is bound to either `:disjunction` or `:conjunction`. Conversely, if `?doc` is bound to itself within the `IF` statement, `FILTER(BOUND(?doc))` evaluates to false.

5.3 Axiomatizing `not` in RDF* and SPARQL*

Having introduced the modeling of $\neg$, $\wedge$, and $\vee$ in the proposed computational ontology, we can now implement the predicates `not`, `and`, and `or`, along with the axioms that infer/constrain the modalities of the involved eventualities based on one another.

We begin with the predicate `not`, which is used more frequently and is a binary predicate (unlike the other two, which are ternary).

The predicate `not` from Hobbs's framework is represented by a corresponding RDF property of the same name in the proposed computational ontology. Both the domain and range of this property are individuals of the class `Eventuality`:

```
:not a rdf:Property; rdfs:domain :Eventuality; rdfs:range :Eventuality.
```

Implementing in SPARQL* the axiom for the `Rexist` modality shown above in (9.a) and repeated again in (49) for reader's convenience, as well as the corresponding axiom for the `Subsist` modality, is similarly straightforward. The SPARQL* rule in (49) again uses the `IF` clause to invert the truth value in the inferred triple and employs the `VALUES` clause to handle both the `Rexist` and `Subsist` modalities.

(49) $\forall_{\text{e, en}}[\texttt{not(en, e)} \rightarrow (\texttt{Rexist(e)} \leftrightarrow \neg\texttt{Rexist(en)})]$

```
[a :InferenceRule; :has-sparql-code """
  CONSTRUCT{?s a ?o}
  WHERE{?e :not|^:not ?ne. VALUES ?m {:Rexist :Subsist}
        BIND(IF(EXISTS{?e a ?m}, <<?ne a ?m>>,
             IF(EXISTS{<<?e a ?m>> a :false}, ?ne, ?s)) AS ?s)
        FILTER(BOUND(?s)) BIND(IF(isTriple(?s), :false, ?m) AS ?o)}"""]
```

The property `not` is also all we need to implement, in SPARQL*, the axioms of the Deontic Traditional Scheme as encoded in Hobbs's logic and shown earlier in (30). These axioms establish constraints among the three deontic modalities. For example, axiom (30.a), repeated in (50) for the reader's convenience, is implemented through the following SPARQL* rule, which closely resembles the one shown in (49):

(50) $\forall_{\text{e,en}}[\texttt{not(en, e)} \rightarrow (\texttt{Permitted(e)} \leftrightarrow \neg\texttt{Obligatory(en)})]$

```
[a :InferenceRule; :has-sparql-code """
  CONSTRUCT{?s a ?o}
  WHERE{?e :not|^:not ?ne. VALUES ?m {:Obligatory :Permitted}
        VALUES ?om {:Obligatory :Permitted} FILTER(?m!=?om)
        BIND(IF(EXISTS?e a ?m,<<?ne a ?om>>,
             IF(EXISTS<<?e a ?m>> a :false,?ne,?s))AS ?s)
        FILTER(BOUND(?s)) BIND(IF(isTriple(?s),:false,?om) AS ?o)}"""]
```

5.4 Axiomatizing `and` and `or` in RDF* and SPARQL*

Unlike the predicate `not`, `and` and `or` are ternary predicates. To represent them in RDF, we chose to introduce *two* RDF properties for each: `and1` and `and2` for `and`, and `or1` and `or2` for `or`. The domains and ranges of these properties are, again, individuals of the class `Eventuality`. Accordingly, `and(ea, e1, e2)` and `or(eo, e1, e2)` are represented in the proposed computational ontology as follows:

```
soa:ea :and1 soa:e1; :and2 soa:e1.
 soa:eo :or1 soa:e1; :or2 soa:e1.
```

Unlike the `not` predicate, implementing the bi-implications that constrain the modalities of eventualities connected by the predicates `and` and `or` requires *two* SPARQL* rules, one for each direction of the bi-implication. This is because the two sides of the bi-implication involve a different number of predicates.

For example, the bi-implication in (9.b) above, which applies to both the `Rexist` and `Subsist` modalities, as well as the `Obligatory` modality since we accept `BDL` axioms `(M)` and `(C)`, as discussed in (32.a) above, is represented by the following SPARQL* rules:

(51) $\forall_{\mathtt{ea,e1,e2}}[\,\mathtt{and(ea,\ e1,\ e2)} \rightarrow (\mathtt{M(ea)} \leftrightarrow (\mathtt{M(e1)} \wedge \mathtt{M(e2)}))]$, with $\mathtt{M} \in \{\mathtt{Rexist,\ Subsist,\ Obligatory}\}$

```
[a :InferenceRule; :has-sparql-code """
  CONSTRUCT{?e1 a ?M. ?e2 a ?M}
  WHERE{?ea :and1 ?e1; :and2 ?e2; a ?M.
        VALUES ?M {:Rexist :Subsist :Obligatory}}"""]

[a :InferenceRule; :has-sparql-code """
  CONSTRUCT{?ea a ?M}
  WHERE{?ea :and1 ?e1; :and2 ?e2.  ?e1 a ?M. ?e2 a ?M.
        VALUES ?M {:Rexist :Subsist :Obligatory}}"""]
```

On the other hand, the bi-implication in (9.c), which, unlike the previous one, applies to the `Rexist` and `Subsist` modalities but not to `Obligatory`, is implemented through the following SPARQL* rule:

(52) $\forall_{\mathtt{ea,e1,e2}}[\mathtt{or(eo,\ e1,\ e2)} \rightarrow (\mathtt{M(eo)} \leftrightarrow (\mathtt{M(e1)} \vee \mathtt{M(e2)}))]$, with $\mathtt{M} \in \{\mathtt{Rexist,\ Subsist}\}$

```
[a :InferenceRule; :has-sparql-code """
  CONSTRUCT{<<<<?e1 a ?M>> a :true>> :disjunction <<<<?e2 a ?M>> a :true>>}
  WHERE{?eo :or1 ?e1; :or2 ?e2; a ?M.
        VALUES ?M {:Rexist :Subsist}}"""]

[a :InferenceRule; :has-sparql-code """
  CONSTRUCT{?eo a ?M}
  WHERE{?eo :or1 ?e1; :or2 ?e2.
        {?e1 a ?M}UNION{?e2 a ?M}UNION
          {<<<<?e1 a ?M>> a :true>> :disjunction <<<<?e2 a ?M>> a :true>>}
        VALUES ?M {:Rexist :Subsist}}"""]
```

In the second rule in (50), note that the second `UNION` clause could be replaced by just its third branch, provided an inference rule is added to derive, from either `?e1 a ?M` or `?e2 a ?M`, the triple in the third branch. However, instead of introducing a new ad hoc rule, we prefer to add a three-branch `UNION` clause.

Finally, we provide the SPARQL* implementation of axiom `(DDS)` from `BDL`, as presented in (32.b):

(53) $\forall_{\mathtt{eo,\,e1,\,e2}}[((\mathtt{or(eo,\,e1,\,e2)}\lor\mathtt{or(eo,\,e2,\,e1)})\land\mathtt{not(en2,\,e2)})\rightarrow$
$((\mathtt{Obligatory(eo)}\land\mathtt{Obligatory(en2)})\rightarrow\mathtt{Obligatory(e1)})]$

```
[a :InferenceRule; :has-sparql-code """
  CONSTRUCT{?e1 a :Obligatory}
  WHERE{{?eo :or1 ?e1; :or2 ?e2}UNION{?eo :or1 ?e2; :or2 ?e1}
        ?eo a :Obligatory. ?en2 :not|^:not ?e2. ?en2 a :Obligatory}"""]
```

5.5 Axiomatizing Abstract Eventualities and Instances in RDF* and SPARQL*

To account for Hobbs's distinction between abstract eventualities and instances, the proposed computational ontology defines two classes: `AbstractEventuality` and `Instance`, corresponding to the two Hobbs's homonymous predicates. These classes are disjoint, therefore the ontology includes a rule similar to (45) that infers a contradiction whenever an eventuality is classified as belonging to both.

In addition, in line with the content of the previous section, the ontology specifies that any eventuality associated with either the `Subsist` modality (or its negation) or with any of the deontic modalities (or their negations) must be an abstract eventuality, whereas any eventuality associated with the `Rexist` modality (or its negation) must be an instance. This distinction is enforced through the following SPARQL* rules:

```
(54) [a :InferenceRule; :has-sparql-code """
       CONSTRUCT{?e a :AbstractEventuality}
       WHERE{VALUES ?m {:Subsist :Obligatory :Permitted :Optional}
             {?e a ?m}UNION{<<?e a ?m>> a :false}}"""]

     [a :InferenceRule; :has-sparql-code """
       CONSTRUCT{?e a :Instance}
       WHERE{{?e a :Rexist}UNION{<<?e a :Rexist>> a :false}}"""]
```

Furthermore, the proposed computational ontology stipulates that all arguments of the predicates `not`, `and`, and `or` must be of the same type: they must either all be abstract eventualities or all be instances. This constraint is enforced by the following SPARQL* rule. If one of these three predicates mixes abstract eventualities with instances, all its arguments will be inferred to belong to both classes. However, since these classes are declared disjoint, this will in turn result in a contradiction.

```
(55) [a :InferenceRule; :has-sparql-code """
       CONSTRUCT{?eao a ?C}
       WHERE{VALUES ?C {:AbstractEventuality :Instance}
             VALUES ?P {:not :and1 :and2 :or1 :or2}
             ?e a ?C. {?e ?P ?eao}UNION{?eao ?P ?e}}"""]
```

Finally, the ontology includes a SPARQL* rule that parallels the implication shown above in (18): if two eventualities instantiate the same abstract eventuality but differ in at least one thematic role (specifically, if one is asserted to have a certain value for that role while the other is asserted not to), then the two eventualities are inferred to be linked by the `not` property: if one of the two instantiations exists, the other does not, and vice versa.

```
(56) [a :InferenceRule; :has-sparql-code """
       CONSTRUCT{?e1 :not ?e2}
       WHERE{?e1 :instantiates+ ?ea.  ?e2 :instantiates+ ?ea.
             FILTER(?e1!=?e2) ?tr a :ThematicRole.
             ?e1 ?tr ?v. <<?e2 ?tr ?v>> a :false}"""]
```

Although the examples so far have only illustrated contradictions, such as those among eventualities that exist or do not, the next section will show that the distinction between abstract eventualities and instances is fundamental to also define concepts such as compliance and violation (and, in turn, conflict, drawing from Kelsen's definitions). As stipulated in the proposed computational ontology, deontic modalities apply exclusively to abstract eventualities. These can be either complied with or violated by an instance that really exists and instantiates the abstract eventuality, or by another abstract eventuality that subsists and is instantiated by the one bearing the deontic modality.

5.6 Examples

Before moving on to the next section, we conclude the present one with a few examples of derivations enabled by the SPARQL* rules introduced so far. All examples are available in the GitHub repository, allowing readers to re-run them locally, verify the results, and experiment with new variants if desired.

Let us begin with a simple example illustrating the axiomatization of the predicates `not`, `and`, and `or` with respect to the `Rexist` modality. Consider sentence (57.a), formalized in Hobbs's framework as (57.b) and represented in RDF as (57.c), along with sentence (57.d), formalized in RDF as (57.e).

(57) a. John leaves or John eats and drinks.

b. Rexist(eo) ∧ or(eo, elj, ea) ∧ Leave(elj) ∧ and(ea, eej, edj) ∧ Eat(eej) ∧ Drink(edj) ∧ has-agent(elj, John) ∧ has-agent(eej, John) ∧ has-agent(edj, John)

c.
```
soa:eo a :Rexist; :or1 soa:elj; :or2 soa:ea.
soa:elj a soa:Leave; soa:has-agent soa:John.
soa:ea :and1 soa:eej; :and2 soa:edj.
```

```
soa:eej a soa:Eat; soa:has-agent soa:John.
soa:edj a soa:Drink; soa:has-agent soa:John.
```

d. John does not leaves.

e. `soa:enlj :not soa:elj; a :Rexist.`

From (57.c) and (57.e), the first rule in (52) derives the triple in (58.a), while the rule in (49) infers the triple in (58.b).

(58) a.
```
<< <<soa:elj a :Rexist>> a :true>> :disjunction
<< <<soa:ea a :Rexist>> a :true>>.
```

b. `<<soa:elj a :Rexist>> a :false.`

From (58.a–b), the rule of Disjunctive Syllogism shown above in (48), followed by the rule in (42), which adds to the knowledge graph every triple whose reification belongs to the class `true`, infers the triple in (59.a). From this triple and (57.c), the first rule shown above in (51) infers the two triples in (59.b): the fact that John eats really exists and the fact that John drinks really exists.

(59) a. `soa:ea a :Rexist`

b. `soa:eej a :Rexist.  soa:edj a :Rexist.`

An example similar to the previous one is the Smith argument discussed in (28) above. Here, we consider the simplified version given in sentences (60.a–b), which are formalized in RDF in (60.c–d), respectively.

(60) a. Smith is obliged to eat or drink.

b. Smith is obliged to not eat.

c.
```
soa:eo a :Obligatory; :or1 soa:ees; :or2 soa:eds.
soa:ees a soa:Eat; soa:has-agent soa:Smith.
soa:eds a soa:Drink; soa:has-agent soa:Smith.
```

d. `soa:enes :not soa:ees; a :Obligatory.`

The first rule shown in (52) above does not apply here, as it is defined only for the `Rexist` and `Subsist` modalities. It is well known that the formula `OB(p∨p)↔(OB(p)∨OB(p))` is invalid. Therefore, for example, from "Smith is obliged to eat or drink", it does not follow that "Smith is obliged to eat or Smith is obliged to drink".

The axiom (`DDS`) was specifically introduced in `BDL` to allow the derivation in the Smith argument even without rules that parallel those in (52) for the `Obligatory` modality. The SPARQL* rule corresponding to (`DDS`), shown above in (53), infers the triple in (61) from (60.c–d), enabling us to conclude that Smith is obliged to drink.

(61) `soa:eds a :Obligatory.`

The GitHub repository associated with this paper also includes the variants of the Smith argument examined in (Goble, 2013), namely the Jones, Roberts, and Thomas arguments. The proposed computational ontology supports the correct derivations in these arguments as well, since it implements the same axioms from `BDL`.

Let us now present an example illustrating the relationship between abstract eventualities and their instantiations. Consider the two contradictory sentences in (62.a-b), which are formalized in RDF respectively as shown in (62.c-d).

(62) a. John pays £3 in cash.

b. John does not pay.

c. `soa:epj a :Rexist, soa:Pay; soa:has-agent soa:John;`
`soa:has-object soa:3pounds; soa:has-instrument soa:cash.`

d. `soa:enpja :not soa:epja; rdf:type :Subsist.`
`soa:epja rdf:type soa:Pay; soa:has-agent soa:John.`

The pragmatic implicature shown above in (39) allows us to infer that the eventualities in (62.c-d) belong to the same context; that is, they refer to the same payment. The following rule then infers the triple: the fact that, in this context, John pays £3 in cash instantiates the fact that John pays.

(63) `soa:epj :instantiates soa:epja.`

In parallel, from (62) the axioms on `not` derive that `soa:epja` does not subsist:

(64) `<<soa:epja a :Subsist>> a :false.`

From this, and from the triple `soa:epj a :Rexist` in (62.c), the SPARQL* rule shown above in (44) infers the contradiction:

(65) `<< <<soa:epja a :Subsist>> a :false>> :is-in-contradiction-with`
`<< <<soa:epj a :Rexist>> a :true>>.`

If it is false that John pays, then it is illogical to also assert, in the same context, that John pays £3 in cash.

Another pair of contradictory sentences is presented in (66.a–b); these are respectively formalized in RDF as shown in (66.c–d). These two sentences contradict one another because cash and card are mutually exclusive instruments for making a payment, as encoded by the SPARQL* rules in (66.e) (if the instrument is cash, then it is not card) and (66.f) (if the instrument is card, then it is not cash).

(66) a. John pays in cash.

b. John pays by card.

c. `soa:epjcash a :Rexist, soa:Pay; soa:has-agent soa:John;`
`soa:has-instrument soa:cash.`

d. `soa:epjcard a :Rexist, soa:Pay; soa:has-agent soa:John;`
`soa:has-instrument soa:card.`

e.
```
[a :InferenceRule; :has-sparql-code """
   CONSTRUCT{<<?e soa:has-instrument soa:card>> a :false}
   WHERE{?e a soa:Pay; soa:has-instrument soa:cash}"""].
```

f.
```
[a :InferenceRule; :has-sparql-code """
   CONSTRUCT{<<?e soa:has-instrument soa:cash>> a :false}
   WHERE{?e a soa:Pay; soa:has-instrument soa:card}"""].
```

From (66.c) and (66.d), the second pragmatic implicature shown above in (40) infers that there exists an abstract eventuality instantiated by both `soa:epjcash` and `soa:epjcard` (this abstract eventuality is an anonymous individual newly created by the rule in (40)):

(67)
```
_:b0 a :AbstractEventuality.
soa:epjcash :instantiates _:b0.  soa:epjcard :instantiates _:b0.
```

In parallel, the two rules in (66.e) and (66.f) symmetrically infer that it is not true that the instrument of `soa:epjcash` is card, and that the instrument of `soa:epjcard` is cash:

(68)
```
<<soa:epjcash soa:has-instrument soa:card>> a :false.
<<soa:epjcard soa:has-instrument soa:cash>> a :false.
```

From (67) and (68), the rule in (56) infers the triple `soa:epjcash :not soa:epjcard`, meaning that the two instances cannot really exist simultaneously. Nevertheless, since they do, two symmetric contradictions are again inferred:

(69)
```
<< <<soa:epjcash a :Rexist>> a :true>> :is-in-contradiction-with
<< <<soa:epjcash a :Rexist>> a :false>>.

<< <<soa:epjcard a :Rexist>> a :true>> :is-in-contradiction-with
<< <<soa:epjcard a :Rexist>> a :false>>.
```

As a final example, we present the encoding of a norm, i.e., the one shown in (4.a) above, repeated in (70) for the reader's convenience, and formalized in Hobbs's as in (36) above.

(70) Whoever parks in a parking spot is obliged to pay £3 at the parking meter associated with that spot.

```
[a :InferenceRule; :has-sparql-code """
  CONSTRUCT{?e2 a soa:Pay,:Obligatory; soa:has-agent ?a;
            soa:has-object soa:3pounds; soa:has-recipient ?pm}
  WHERE{?e1 a soa:Park,:Rexist; soa:has-agent ?a; soa:has-location ?p.
        ?p a soa:parkingSpot. ?pm soa:associated-with ?p
        BIND(BNode() AS ?e2) NOT EXISTS{?er2 a soa:Pay,:Obligatory;
        soa:has-agent ?a; soa:has-object soa:3pounds;
        soa:has-recipient ?pm}}"""]
```

Since the consequent of the implication involves an existential quantification, similarly to the SPARQL* rule in (40) the rule in (70) generates a new anonymous RDF resource, corresponding to the Skolemization of the existential quantifier. Additionally, the rule includes a `NOT EXISTS` clause to prevent infinite loops.

Now, if the state of affairs includes the triples in (71.b), which encodes sentence (71.a), then the rule in (70) adds to the knowledge graph the triples in (71.c), in which `_:b0` is again an anonymous individual: John is obliged to pay £3 to the parking meter associated with the spot in Sketty where he parked.

(71) a. John parks in Sketty.

b.
```
soa:epkj a soa:Park, :Rexist; soa:has-agent soa:John;
soa:has-location soa:psSketty.  soa:psSketty a soa:parkingSpot.
soa:pmSketty soa:associated-with soa:psSketty.
```

c.
```
_:b0 a :Obligatory, soa:Pay; soa:has-agent soa:John;
soa:has-object soa:3pounds; soa:has-recipient soa:pmSketty.
```

6 Incorporating Kelsen's categorization of irresolvable conflicts into the proposed framework

With the basic framework now established, this section presents our proposal for incorporating irresolvable conflicts among deontic statements.

This paper adopts Kelsen's categorization of irresolvable conflicts, as outlined in Section 2 above. We recall that Kelsen identifies three primary types of irresolvable conflicts:

(72) a. **Bilateral conflicts**: a situation in which two deontic statements hold in a given context, but complying with one of them entails violating the other.
Example:
- John is obliged to leave the building.
- John is obliged to not leave the building.

b. **Unilateral conflicts**: a situation in which an obligation and a permission hold in a given context, but acting in accordance with the permission entails a violation of the obligation.
Example:
- John is permitted to leave the building.
- John is prohibited to leave the building.

c. **Partial conflicts**: a type of bilateral conflict where the conflict arises only from a "part" of the actions involved, or, according to this paper's terminology, due to at least one thematic role having incompatible values.
Example:
- John is obliged to pay in cash.
- John is obliged to pay by card.

The review in (Goble, 2013) addresses only the category in (72.a), but it remains unclear to us how the reviewed approaches could be extended to encompass (72.b) or (72.c). Moreover, even for the category in (72.a), (Goble, 2013) adopts a definition of irresolvable conflicts that, in our view, is not fully adequate. We will examine Goble's definition in detail and compare it with Kelsen's in Subsection 6.3 below.

The key characteristic of Kelsen's definitions in (72.a-c), which is absent in Goble's, is that they are grounded in the notion of *violation*: a conflict arises when an agent does what he is obliged or permitted to do, but this action *violates* another of his obligations.

Despite its close connection to the notions of obligation and prohibition, the concept of violation, and its corresponding formalization, has been largely overlooked in previous deontic logic literature. This paper argues, instead, that violation should be taken as a foundational concept. For this reason, the next subsection introduces and formalizes the notion of violation, along with the dual notion of compliance, within the proposed computational ontology. We will then build on this foundation to identify and formalize the notion of conflict between deontic statements.

6.1 Understanding and formalizing compliance and violations

Obligations and prohibitions (i.e., non-permissions) can be either complied with or violated. This may occur when the bearer of an obligation or prohibition performs an action that *specializes* the one specified by the obligation or prohibition. In the proposed computational ontology, this is represented by requiring the knowledge graph to contain an eventuality that *really exists* and that *instantiates* the abstract eventuality to which `Obligatory` or ¬`Permitted` applies. Specifically, if the really existing eventuality instantiates an obligatory abstract eventuality, it *complies with* it; if it instantiates a prohibited abstract eventuality, it *violates* it. This is illustrated in Figure 2, where the dotted line denotes the `instantiates` property and the solid line indicates the inferred statement.

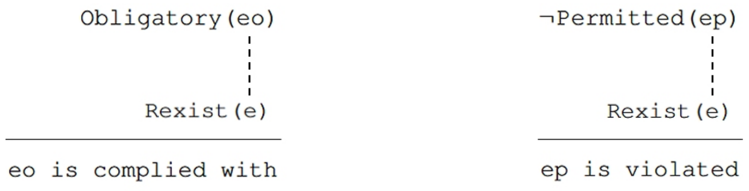

Figure 2: Schema illustrating compliance with obligations and violation of prohibitions. The dotted line represents the `instantiates` property while the solid line indicates the inferred statement.

(73) presents three brief example sentences illustrating the rationale behind the schema in Figure 2, along with their corresponding RDF representations:

(73)
a. John is obliged to pay £3:
```
soa:eo a Obligatory; soa:has-agent soa:John;
soa:has-object soa:3pounds.
```
b. John is not permitted to pay in cash.
```
<<soa:ep a Permitted>> a :false.  soa:ep a soa:Pay;
soa:has-agent soa:John; soa:has-instrument soa:cash.
```
c. John pays £3 in cash:
```
soa:e a :Rexist; soa:has-agent soa:John; soa:has-object
soa:3pounds; soa:has-instrument soa:cash.
```

The sentence in (73.c) complies with (73.a) but violates (73.b). To comply with (73.a) without violating (73.b), John would have needed to make the payment by card or any means other than cash. Indeed, note that (73.a) does not specify any instrument for making the payment: John can choose whichever he wishes. Similarly, (73.b) prohibits John from paying any amount in cash, whether £3, £4, or otherwise.

Regarding the corresponding (RDF) formal representations, as illustrated in the examples from the previous section, the first pragmatic implicature shown in (39) allows us to infer that `e` instantiates both `eo` and `ep`.

Now, in order to infer that the fact that `eo` is obligatory is complied with by the fact that `e` really exists, while the fact that `ep` is not permitted is violated by the fact that `e` really exists, we need to introduce SPARQL* rules similar to the one shown above in (43), which allow us to infer when a fact `is-in-contradiction-with` another one.

To this end, we introduce in the proposed computational ontology the following two SPARQL* rules, along with the properties `is-complied-with-by` and `is-violated-by`. These rules implement the schema shown in Figure 2.

(74)
```
[a :InferenceRule; :has-sparql-code """
  CONSTRUCT{<< <<?eo a :Obligatory>> a :true>> :is-complied-with-by
            << <<?e a ?m>> a :true>>}
  WHERE{?eo a :Obligatory.  ?e a ?m; :instantiates+ ?eo.
        VALUES ?m {:Rexist :Subsist}}"""]

[a :InferenceRule; :has-sparql-code """
  CONSTRUCT{<< <<?ep a :Permitted>> a :false>> :is-violated-by
            << <<?e a ?m>> a :true>>}
  WHERE{<<?ep a :Permitted>> a :false.  ?e a ?m; :instantiates+ ?ep.
        VALUES ?m {:Rexist :Subsist}}"""]
```

It is worth noting that the rules in (74) check whether `?e` holds under either the `Rexist` or the `Subsist` modality. Therefore, even when `?e` is an abstract eventuality that partially instantiates `?eo` (or `?ep`), it is still inferred that the former complies with (or violates) the latter. This is because, as explained in Subsection 3.2 above, whenever an abstract eventuality subsist, at least one of its full instantiations really exists. Since `instantiates` is a transitive relation, if in (74) `?e` is an abstract eventuality that subsists and instantiates `?eo`, then each of `?e`'s full instantiations (including the one that really exists) also instantiates `?eo`, and the schema in Figure 2 applies again.

The key difference between the examples presented in the previous section, where `is-in-contradiction-with` is inferred, and those involving the rules in (74), where `is-complied-with-by` and `is-violated-by` are inferred, lies in the modalities involved. When both modalities indicate existence, i.e., `Rexist` or `Subsist`, a contradiction is inferred. On the other hand, when the more abstract eventuality indicates a deontic modality, specifically, an obligation or a non-permission, and the modality instantiating that eventuality indicates existence, compliance or violation is inferred. In the next subsection, we will see that conflicts, instead, isomorphically involve two deontic modalities: when two facts are both obligatory and non-obligatory, or both permitted and non-permitted, a conflict is inferred.

Two additional symmetric rules are introduced to infer when a prohibition is complied with and when an obligation is violated. This inference is possible when the knowledge graph includes an eventuality that is more abstract than the obligatory and the non-permitted ones, and this eventuality does not subsist, as illustrated in Figure 3.

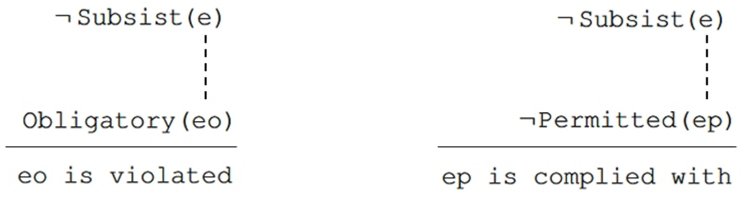

Figure 3: Schema illustrating violation of obligations and compliance with prohibitions. The dotted line represents the `instantiates` property while the solid line indicates the inferred statement.

The two rules are the following:

```
(75) [a :InferenceRule; :has-sparql-code """
       CONSTRUCT{<< <<?eo a :Obligatory>> a :true>> :is-violated-by
                 << <<?e a :Subsist>> a :false>>}
       WHERE{?eo a :Obligatory.  ?eo :instantiates+ ?e.
             <<?e a :Subsist>> a :false}"""]

     [a :InferenceRule; :has-sparql-code """
       CONSTRUCT{<< <<?ep a :Permitted>> a :false>> :is-complied-with-by
                 << <<?e a :Subsist>> a :false>>}
       WHERE{<<?ep a :Permitted>> a :false.  ?ep :instantiates+ ?e.
             <<?e a :Subsist>> a :false}"""]
```

The next subsubsection provides additional examples illustrating the four rules in (74) and (75). We remind the reader once again that all examples have been implemented and are available in the GitHub repository associated with this paper.

6.1.1 Examples

(73) above has already shown some simple examples of the rules in (74). Another example is the following:

(76)
a. John is prohibited to not pay.

b. `<<soa:enpj a :Permitted>> a :false.  soa:enpj :not soa:epj.`
   `soa:epj a soa:Pay; soa:has-agent soa:John.`

c. John pays £3.

d. `soa:epj3 a :Rexist, soa:Pay; soa:has-agent soa:John.`

This example differs from the previous ones because the rules in (74) do not apply directly to the RDF representations in (81.b) and (81.d). Instead, first the rules from the Deontic Traditional Scheme apply to derive that, since John is prohibited from not paying, he is obliged to pay. Specifically, the SPARQL* rule in (50) derives (77.a) from (81.b). Then, from (77.a) and (81.d), the first rule in (74) infers (77.b): the fact that John is obliged to pay is complied with by the fact that he pays £3.

(77)
a. `soa:epj a :Obligatory.`

b. `<< <<epj a :Obligatory>> a :true>> :is-complied-with-by`
   `<< <<epj3 a :Rexist>> a :true>>`

Let us now remind the reader that the two rules in (74) also accept the modality `Subsist` for the eventuality that instantiates the obligatory or non-permitted eventuality. However, all examples presented so far in this subsection feature only the `Rexist` modality for the instantiating eventuality.

Nevertheless, it is indeed rather difficult to construct clearly understandable examples involving non-negated abstract eventualities in the *present* tense. Therefore, let us instead consider some examples in the *past* tense. Suppose, for instance, that we knew John had £100 in cash yesterday, but now he has none. When asked about it, he replied that he spent it all yesterday. From this, we might deduce that:

(78) Yesterday, John paid in cash.

```
soa:eay a :Subsist, soa:Pay; soa:has-agent soa:John;
soa:has-time soa:Y; soa:has-instrument soa:cash.
```

The formal representation in (78) introduces a new thematic role not previously considered: `soa:has-time`. The value of `soa:has-time` is '`soa:Y`', which stands for "yesterday". This is the only example in the paper that includes this thematic role, because time management is regarded as an area for future research, as discussed in Section 8.

Apart from this, `eay` in (78) is an *abstract* eventuality in that it does not refer to a specific action of paying that took place yesterday. Conversely, since John spent his £100 in cash, (78) indicates that John made some cash payments yesterday amounting to a total of £100. While the exact number of payments is unknown (nor is it important for the meaning that the sentence in (78) intends to convey), we do know there was *at least one*; i.e., in the extreme case, John spent all of his £100 in a single payment.

Assuming that the prohibition in (73.b) was also in force yesterday, i.e., that it holds:

(79) Yesterday, John was not permitted to pay in cash.
```
<<soa:epy a :Permitted>> a :false.  soa:epy a soa:Pay;
soa:has-agent soa:John; soa:has-time soa:Y; soa:has-instrument soa:cash.
```

From (78) and (79), the second rule in (74) infers that the fact that `eay` subsists violates the fact that `epy` is not permitted, i.e.:

(80)
```
<< <<epy a :Permitted>> a :false>> :is-violated-by
<< <<eay a :Subsist>> a :true>>
```

As an example of the two rules in (75), which are symmetrical to those in (74), we need to consider abstract eventualities that do not subsist, such as:

(81)

a. John does not pay.

b.
```
<<soa:ejnp a :Subsist>> a :false.  soa:ejnp a soa:Pay;
soa:has-agent soa:John.
```

c. John is obliged to pay £3 in cash.

d.
```
soa:eopj3c a :Obligatory, soa:Pay; soa:has-agent soa:John;
soa:has-object soa:3pounds; soa:has-instrument soa:cash.
```

If John does not pay, then he fails to pay any amount of money by any means, thereby violating his obligation in (81.c). This is inferred by the first rule in (75):

(82)
```
<< <<eopj3c a :Obligatory>> a :true>> :is-violated-by
<< <<ejnp a :Subsist>> a :false>>
```

6.2 Understanding and formalizing conflicts among deontic statements

The previous subsection introduced the notion of violations of obligations and prohibitions and demonstrated how to formalize it within the proposed computational ontology.

The rationale behind the rules introduced in the previous subsubsection can be illustrated as shown in Figure 4, which again refers to the two examples in (73.a) and (73.b). Each obligation is associated with a "green area" representing the *set* of all really existing eventualities that instantiate the obligatory eventuality; these eventualities comply with the obligation. For example, in Figure 4.a, the obligation in (73.a) is complied with by any eventuality that really exists and specifies the same thematic roles as the obligation (possibly including additional thematic roles, e.g., the instrument). Conversely, each prohibition is associated with a "red area" representing the *set* of all really existing eventualities that instantiate the prohibited eventuality; these eventualities violate the prohibition. For instance, in Figure 4.b, the prohibition in (73.b) is violated by any eventuality that really exists and specifies the same thematic roles as the prohibition (possibly including additional thematic roles, e.g., the object).

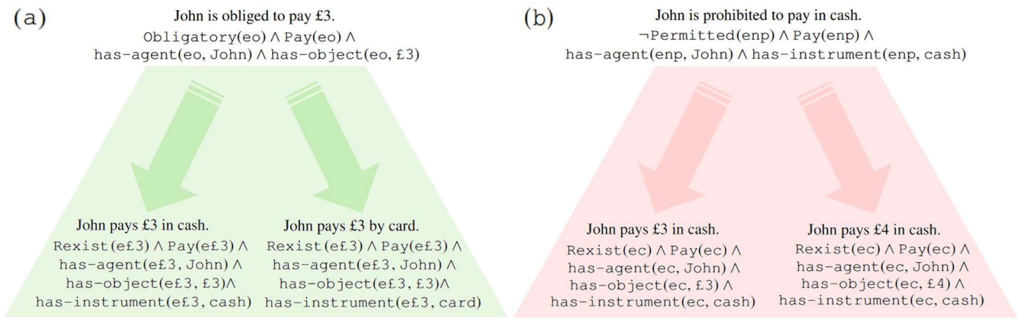

Figure 4: Compliance of the obligation (73.a) and violations of the prohibition (73.b)

To understand and formalize conflicts, we need to consider two eventualities that *both* hold for specific deontic modalities, such as an obligation and a prohibition.

When a prohibited eventuality instantiates an obligatory one, meaning it specifies the same action or state and the same thematic roles with identical values as the latter, no conflict occurs. In such a context, it is indeed possible to select an eventuality within the green area but outside the red area. In other words, there is an eventuality that complies with the obligation without violating the prohibition. For example, in Figure 5.a, John paying £3 *by card* complies with his obligation to pay £3 and does not violate his prohibition against paying £3 *in cash*.

On the other hand, when an obligatory eventuality instantiates a prohibited one, the red area includes the green area, making it impossible to find an eventuality that complies with the obligation without also violating the prohibition. This situation constitutes a conflict: any eventuality that complies with the obligation also violates the prohibition. An example appears in Figure 5.b, where the only way for John to comply with his obligation to pay £3 *in cash* is to violate his prohibition against using cash for payments. The same logic applies to permissions, although permissions are not "complied with" or "violated" but rather "applied" or "denied". Apart from this difference in terminology, permissions behave like obligations in relation to conflicts. For instance, as shown in Figure 5.c, if John performs a more specific action allowed by his permission to pay £3 in cash, he again violates his prohibition against paying in cash.

Since, according to the Deontic Traditional Scheme, obligations entail permissions (in symbols: $\texttt{OB(p)}\rightarrow\texttt{PE(p)}$), both configurations in Figure 5.b and Figure 5.c, which correspond respectively to Kelsen's bilateral and unilateral conflicts (see (72) above), can be addressed using a single SPARQL* rule, shown in (83). This rule searches the knowledge graph for the pattern $\texttt{PE(p)}\wedge\neg\texttt{PE(q)}$, where `p` denotes an eventuality that instantiates the one denoted by `q`. If this pattern is found, the rule asserts that the two statements

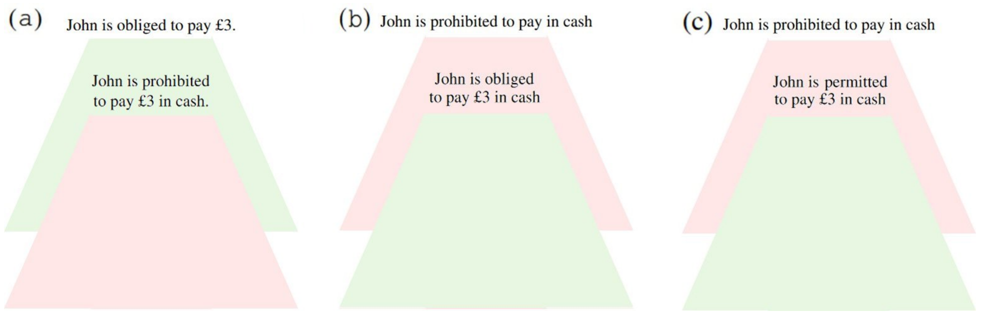
(a)
John is obliged to pay £3.
John is prohibited to pay £3 in cash.
(b)
John is prohibited to pay in cash
John is obliged to pay £3 in cash
(c)
John is prohibited to pay in cash
John is permitted to pay £3 in cash

Figure 5: General schema of conflicts between deontic statements.

are in conflict. This is done by connecting the reifications of the two statements through the RDF property `is-in-conflict-with`.

(83)
```
[a :InferenceRule; :has-sparql-code """
  CONSTRUCT{<< <<?ep a :Permitted>> a :true>> :is-in-conflict-with
            << <<?enp a :Permitted>> a :false>>}
  WHERE{?ep a :Permitted; :instantiates+ ?enp.
        <<?enp a :Permitted>> a :false}"""]
```

6.2.1 Examples

This subsubsection presents examples of inferred conflicts using the SPARQL* rule in (83). As a first example, consider sentences (84.a) and (84.c), which are formalized in RDF as shown in (84.b) and (84.d), respectively.

(84)

a. It is optional for John to leave the building.

b.
```
soa:elj a soa:Leave,:Optional; soa:has-agent soa:John;
soa:has-from-location soa:B; :not soa:enlj.
```

c. John is not permitted to leave the building.

d.
```
<<soa:elpj a :Permitted>> a :false.  soa:elpj a soa:Leave;
soa:has-agent soa:John; soa:has-from-location soa:B.
```

The SPARQL* rules corresponding to the axioms of the Deontic Traditional Scheme, shown above in (30), infer from (84.a) that John is not obliged to leave the building and that he is not obliged to not leave the building, i.e., (85.a). From this, they further infer that John is permitted to leave the building and that he is permitted to not leave the building, i.e., (85.b). Finally, from (85.b) and (84.b), the SPARQL* rule in (83) infers that the fact that John is not permitted to leave the building is in conflict with the fact that he is permitted to leave the building, encoded as in (85.c).

(85)
a. `<<soa:elj a :Obligatory>> a :false.`
   `<<soa:enlj a :Obligatory>> a :false.`

b. `soa:elj a :Permitted.  soa:enlj a :Permitted.`

c. `<<<<soa:elj a :Permitted>> a :true>> :is-in-conflict-with`
   `<<<<soa:elpj a :Permitted>> a :false>>.`

Let us now present an example of partial conflict, which corresponds to the third category of conflicts defined by Kelsen. Consider the two conflicting obligations in (86.a–b), which represent a "deontic variant" of the two contradictory sentences shown in (66.a–b) above:

(86)
a. John is obliged to pay in cash.

b. John is obliged to pay by card.

(86.a–b) are represented in RDF as follows. The only difference between (87.a) and (87.b) lies in the instrument used for the action of paying: cash in one case, card in the other.

(87)
a. `soa:eopjcash a :Obligatory, soa:Pay; soa:has-agent soa:John;`
   `soa:has-instrument soa:cash.`

b. `soa:eopjcard a :Obligatory, soa:Pay; soa:has-agent soa:John;`
   `soa:has-instrument soa:card.`

As in the examples in (66.a–b), the second pragmatic implicature shown above in (40) infers the existence of an (anonymous) abstract eventuality that both `soa:eopjcash` and `soa:eopjcard` instantiate. In other words, `soa:eopjcash` and `soa:eopjcard` are inferred to be two different "variants" of the same abstract act of paying that John is obliged to perform. This is represented in RDF as follows:

(88)
```
_:b0 a :AbstractEventuality.
soa:eopjcash :instantiates _:b0.  soa:eopjcard :instantiates _:b0.
```

As in the examples in (66.a–b), from (87) and (88), the two SPARQL* rules in (66.e) and (66.f), which state that cash and card are mutually exclusive instruments for paying, together with the rule in (56), which specifies that eventualities instantiating the same abstract eventuality and differing on mutually exclusive values for the same thematic role are connected by the `not` property, infer the following triple:

(89) `soa:eopjcash :not soa:eopjcard`

From this, the SPARQL* rules implementing the Deontic Traditional Scheme infer that `soa:eopjcash` and `soa:eopjcard` are both permitted and not permitted. From this, the rule in (83) in turn infers the following two symmetric conflicts:

(90)

a. `<<<<soa:eopjcash a :Permitted>> a :true>> :is-in-conflict-with <<<<soa:eopjcash a :Permitted>> a :false>>.`

b. `<<<<soa:eopjcard a :Permitted>> a :true>> :is-in-conflict-with <<<<soa:eopjcard a :Permitted>> a :false>>.`

6.3 Comparing the proposed solution with the conflict-tolerant deontic logics reviewed in (Goble, 2013)

In Subsection 4.1, we reviewed the principal approaches proposed in the literature to address irresolvable normative conflicts. These approaches are systematically categorized by Goble in his seminal review (Goble, 2013) into three main strategies: (1) Revisionist strategies, (2) Paraconsistent deontic logics, and (3) Other radical strategies. (Goble, 2013) remains one of the most comprehensive and influential surveys in the field for researchers studying deontic conflicts.

In addition to categorizing conflict-tolerant strategies, Goble systematically evaluates individual approaches within each of the three categories by examining how well they satisfy three central desiderata. These desiderata, articulated in (Goble, 2013), reflect key logical and conceptual requirements for any deontic system intended to handle irresolvable normative conflicts. Desideratum #1 demands that the logic treat conflicting deontic statements as *consistent*, recognizing that agents may face genuine normative dilemmas without producing illogical conclusions. Desideratum #2 seeks to *avoid deontic explosion*, ensuring that the presence of conflict does not trivialize the system by implying that everything is obligatory. Desideratum #3 requires that the logic *preserve intuitive deontic reasoning*, offering plausible accounts of commonly accepted normative inferences. Goble's comparison highlights the trade-offs faced by different approaches and underscores the difficulty of satisfying all three desiderata simultaneously.

The present subsection sheds some light on the similarities and differences between the approach proposed in this paper and the three main strategies categorized by Goble, in light of the three desiderata.

Each of these strategies is discussed in detail in the corresponding subsubsections of Subsection 4.1. Their rationale could be depicted as in Figure 6.

All three approaches build on Standard Deontic Logic (SDL), which extends classical modal logic by introducing a deontic operator, `OB`, whose inferences are governed by a specific axiomatization. However, SDL is unable to represent irresolvable conflicts.

To address this limitation, each of the three strategies modifies the underlying infrastructure of SDL in a different way. Revisionist strategies, as indicated by the red arrow, revise the axiomatization of the `OB` operator itself; in other words, they operate at the second level by exploring alternative axioms that avoid problematic inferences such as deontic explosion (Desideratum #2) while still supporting intuitive ones, such as the Smith argument (Desideratum #3). Paraconsistent deontic logics, by contrast, achieve this by intervening at the level of classical logic, specifically targeting the principle of Ex falso quodlibet, which is responsible for the explosive behavior when contradictions arise. Finally, other radical strategies typically preserve the first two levels but introduce an additional layer whose constructs govern the reasoning process over the base logic to

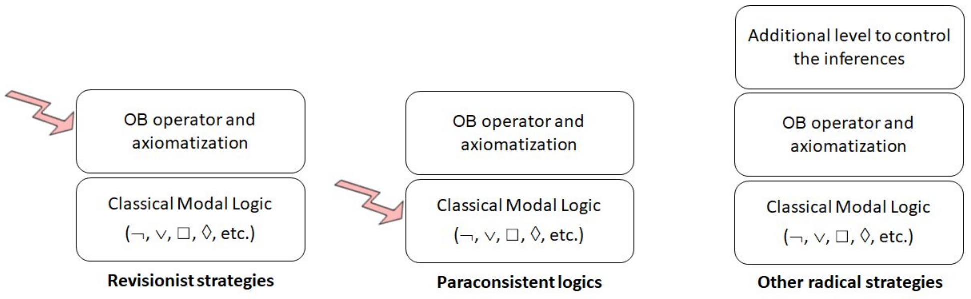

Figure 6: A graphical comparison of conflict-tolerant deontic logics proposed in the literature

satisfy the three desiderata. This additional layer may enforce a specific rule execution order (as in Two-phase Deontic Logic), represent commands associated with obligations (as in imperatival approaches), or encode abnormalities along with strategies for handling them (as in Adaptive Deontic Logics).

The main technical difference between the three types of conflict-tolerant deontic logics depicted in Figure 6 and the computational ontology proposed in this paper concerns the structure of the underlying levels. As illustrated in Figure 7, the proposed ontology *swaps* the first two levels. Modalities, such as `Rexist` and the deontic modalities (e.g., `Obligatory`), are placed at the lowest level. The operators of classical modal logic, including Boolean connectives such as $\neg$ and $\vee$, and as will be shown in the next section, the modal operators $\Box$ and $\Diamond$, are then implemented above this. The third level represents abnormalities, including contradictions and conflicts, which are encoded by the RDF properties `is-in-contradiction-with` and `is-in-conflict-with` respectively.

Another important technical difference is that, due to the use of *semantical embedding*, all three levels are implemented using the same formal languages: RDF* and SPARQL*. These are Semantic Web standards, which makes the ontology suitable for practical implementation and wide-scale industrial deployment, as discussed in the Introduction. In contrast, to the best of our knowledge the approaches reviewed in (Goble, 2013) have not been implemented and remain at a purely theoretical level.

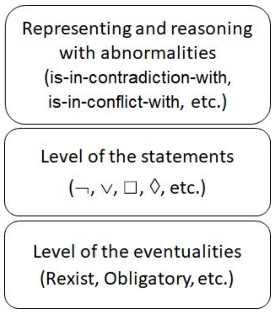
Representing and reasoning with abnormalities
(is-in-contradiction-with, is-in-conflict-with, etc.)
Level of the statements
(¬, ∨, □, ◊, etc.)
Level of the eventualities
(Rexist, Obligatory, etc.)

Figure 7: The three levels of the computational ontology proposed in this paper

On the other hand, the proposed computational ontology also shares insights with each of the three strategies of conflict-tolerant deontic logics depicted in Figure 6 and, in some sense, can be seen as a solution that mediates among all three.

The SPARQL* rules defined for handling the meaning of the deontic modalities closely parallel the axiomatization of the `BDL` logic discussed in Subsubsection 4.1.1 above. This logic is one of the only two revisionist approaches reviewed in (Goble, 2013) that successfully meet all three desiderata (the other being a slight variant of `BDL`). The only minor difference between the SPARQL* rules proposed here and the `BDL` axiomatization lies in the `BDL`'s `RBE` axiom, which essentially states that if two formulas are equivalent and one is obligatory, then the other is also obligatory. Our formalization uses the standard bi-implication ($\leftrightarrow$) to represent logical equivalence, whereas `RBE` imposes additional constraints on bi-implication, thereby restricting the standard notion of logical equivalence (see (Robaldo and Liga, 2025) for further details).

The additional constraints implemented by `RBE` are necessary to avoid deontic explosion in `BDL`. In contrast, the proposed computational ontology achieves this by following the approach of paraconsistent logics: deontic explosion is prevented by avoiding SPARQL* rules that would lead to the Ex falso quodlibet. Specifically, this paper chooses not to implement SPARQL* rules corresponding to the rule of Disjunctive Introduction, which, together with the rule of Disjunctive Syllogism, lead to Ex falso quodlibet.

Finally, the proposed computational ontology also shares a key feature with Adaptive Deontic Logics, one of the radical strategies reviewed in (Goble, 2013). Specifically, Adaptive Deontic Logics is the only conflict-tolerant deontic logic that explicitly represents *abnormalities*, such as conflicts, whereas all other logics merely treat conflicting obligations as consistent formulae, indistinguishable from other consistent statements. In our proposed computational ontology, contradictions and conflicts are inferred as new triples using the RDF properties `is-in-contradiction-with` and `is-in-conflict-with`, enabling their detection through a simple SPARQL query over the inferred knowledge graph. Moreover, as will be clarified in the next section, it is both possible and arguably desirable to introduce a broader range of abnormalities beyond contradictions and conflicts, to capture more nuanced meanings. To implement this within the proposed ontology, it suffices to add RDF properties isomorphic to `is-in-contradiction-with` and `is-in-conflict-with`, along with corresponding SPARQL* inference rules.

On the other hand, there are also *more fundamental, non-technical* differences between the proposed computational ontology and the conflict-tolerant deontic logics reviewed in (Goble, 2013), specifically regarding differing intuitions about the notion of irresolvable conflict. We consider these differences in intuition to be the main contribution of this work to the ongoing research in deontic logic.

The conflict-tolerant deontic logics reviewed in (Goble, 2013) are based on the intuition and definition given in Desideratum #1: a conflict (of obligations) is described as a situation in which "an agent ought to do a number of things, each of which is possible for the agent, but it is impossible for the agent to do them all".

(Goble, 2013) explicitly formalizes this notion as $\mathtt{OB(A)} \wedge \mathtt{OB(B)} \wedge \neg\Diamond(\mathtt{A} \wedge \mathtt{B})$ and employs this formula in derivations to assess whether the various conflict-tolerant deontic logics reviewed meet the three desiderata.

In this paper, we argue that Goble's definition captures an important subclass of normative conflicts but does not generalize to the broader range of conflicts that arise when permissions, contextual constraints, and violations are taken into account.

In particular, rather than modeling genuine clashes between obligations and/or permissions, Goble's definition addresses the relationship between what an agent *must* do and what an agent *can* do. That is, it focuses on the interaction between deontic prescriptions and the *contextual constraints* governing the agent's practical possibilities. From this perspective, Goble's definition is more concerned with feasibility than with normativity in the stricter legal-theoretical sense.

By contrast, the present work adopts an alternative but compatible intuition inspired by Kelsen's legal theory, in which obligations and prohibitions are defined independently of what is factually possible. In this view, the central notion is not feasibility but *violation*: a conflict arises when fulfilling one obligation, or acting in accordance with what is permitted, necessarily leads to the violation of another.

This alternative perspective does not contradict Goble's account but rather extends it by addressing a broader class of normative phenomena. Whereas Goble highlights conflicts rooted in practical limitations, our framework allows for modeling conflicts that arise from the internal structure of normative systems, even in idealized settings where all actions are feasible. Both perspectives can be naturally accommodated within the proposed ontology, provided that contextual constraints are explicitly represented.

To account for situations involving contextual constraints on action, our framework must be extended to represent what is *possible* or *necessary* for an agent within a given state of affairs. The next section presents how these modal concepts are modeled in the proposed computational ontology, thereby extending its expressive power to encompass both the normative and the practical dimensions of agency.

7 Deontic modalities and contextual constraints

As mentioned earlier, (Goble, 2013) adopts the following formal definition of conflicts among obligations to determine whether the conflict-tolerant deontic logics proposed in the literature satisfy the three desiderata: $\mathtt{OB}(\mathtt{A}) \wedge \mathtt{OB}(\mathtt{B}) \wedge \neg\Diamond(\mathtt{A} \wedge \mathtt{B})$.

As explained above, we chose not to use this formal definition, as we believe it concerns the interaction between obligation and possibility, rather than the interaction between two obligations. In fact, this definition also appears to hold in cases involving a single obligation, such as $\mathtt{OB}(\mathtt{C}) \wedge \neg\Diamond(\mathtt{C})$. Consider again the sentences previously shown in (4.b) and (5), which are repeated in (91.a-b) for the reader's convenience.

(91) a. The parking meter in Sketty only accepts cash.

b. It is prohibited to pay in cash.

(91.a) denotes a *constraint* holding in the state of affairs: in Sketty, it is *not possible* to avoid paying in cash, i.e., it is *necessary* to pay in cash. In standard propositional modal logic, this could be represented as $\neg\Diamond(\neg\mathtt{C})$. (91.b) states that paying in cash is prohibited, i.e., it is obligatory not to pay in cash. In standard propositional deontic logic, this could be represented as $\mathtt{OB}(\neg\mathtt{C})$. Therefore, whenever $\mathtt{OB}(\mathtt{A} \wedge \mathtt{B}) \leftrightarrow \mathtt{OB}(\mathtt{A}) \wedge \mathtt{OB}(\mathtt{B})$ holds, as in BDL and in the proposed computational ontology, the definition used in (Goble, 2013) corresponds to a specific case of what is exemplified in (91): when $\mathtt{A} \wedge \mathtt{B} \leftrightarrow \neg\mathtt{C}$.

Note that the prohibition in (91.b) will be *violated*. However, this violation does not occur because its bearers comply with another obligation or prohibition, but rather because of the constraint present in the state of affairs, i.e., due to what the prohibition's bearers *can* do in the given context, as discussed above.

This constitutes an abnormality different from those modeled in the previous section and should therefore correspond, in the proposed computational ontology, to a different RDF property: the bearers of (91.b) will *necessarily* violate the prohibition.

Building on the foregoing, this section extends the proposed computational ontology to capture the interplay between deontic statements and contextual constraints.

Two new classes are introduced in the ontology's vocabulary: `necessary` and `possible`, which serve as alternatives to `true` and `false`:

(92)
```
:necessary a rdfs:Class; rdfs:subClassOf rdf:Statement.

:possible a rdfs:Class; rdfs:subClassOf rdf:Statement.
```

In parallel, we observe that contextual constraints seem to apply only to the *thematic roles* of an eventuality. In other words, they restrict *how* certain actions or states may occur. For example, (91.a) constrains the *instrument* used in parking payments made in Sketty: only cash is permitted. Therefore, by using these classes, the fact that it is *necessary* for the instrument of the paying action `ep` to be cash is encoded as in (93.a) while the fact that this is *possible* is encoded as in (93.b):

(93)
```
a. <<soa:ep soa:has-instrument soa:cash>> a :necessary.

b. <<soa:ep soa:has-instrument soa:cash>> a :possible.
```

In Hobbs's framework, (93.a) may be represented as $\Box$`soa:has-instrument(ep, cash)`, while (93.b) as $\Diamond$`soa:has-instrument(ep, cash)`. Since RDF does not provide operators corresponding to $\Box$ and $\Diamond$, the classes in (92) have been introduced in a manner analogous to the handling of standard negation $\neg$.

As is well known, the equivalence $\Box A \leftrightarrow \neg\Diamond\neg A$ holds. This means that if something is necessary, then its negation is not possible, and vice versa. However, the SPARQL* rules implementing this equivalence are not included in this paper, as they are not needed for the examples presented below. As with the distributive laws for conjunction and disjunction or De Morgan's laws, their implementation is left to the reader as an exercise.

One well-known property of $\Box$ must instead be incorporated into the proposed computational ontology to support the correct derivations in the examples that follow: the axiom $\Box A \rightarrow A$, which, in standard Kripke semantics, corresponds to reflexivity of the accessibility relation. The SPARQL* rule expressing this axiom is shown in (94) and asserts as true every statement that is necessary.

(94)
```
[a :InferenceRule; :has-sparql-code """
    CONSTRUCT{?s ?p ?o}WHERE{<<?s ?p ?o>> a :necessary}"""]
```

(91.a) can now be represented using the SPARQL* rule in (95), which states that every payment made at the parking meter associated with the parking spot in Sketty was necessarily carried out using cash.

(95)
```
[a :InferenceRule; :has-sparql-code """
   CONSTRUCT{<<?ep soa:has-instrument soa:cash>> a :necessary}
   WHERE{?ep a soa:Pay; soa:has-recipient ?pm.
         ?pm soa:associated-with soa:psSketty}"""]
```

Now, by encoding that John pays at the parking meter linked to the parking spot in Sketty, that is, by asserting the RDF triples in (96):

(96)
```
soa:epsj a soa:Pay,:Rexist; soa:has-agent soa:John;
soa:has-recipient soa:pmSketty.  soa:psSketty a soa:parkingSpot.
soa:pmSketty soa:associated-with soa:psSketty.
```

From (95), it is inferred that it is necessary for the instrument of `soa:epsj` to be cash. Then, based on (94), it follows that the instrument of `soa:epsj` is in fact cash. This is represented by the following triples:

(97)
```
<<soa:epsj soa:has-instrument soa:cash>> a :necessary.
soa:epsj soa:has-instrument soa:cash.
```

Let us now complete the example in (91) by adding the representation of the prohibition stated in (91.b). This corresponds to the following SPARQL* rule:

(98)
```
[a :InferenceRule; :has-sparql-code """
   CONSTRUCT{<<?eppca a :Permitted>> a :false.
      ?eppca a soa:Pay; soa:has-agent ?a; soa:has-instrument soa:cash}
   WHERE{?e soa:has-agent ?a.  BIND(BNode() AS ?eppca)
      NOT EXISTS{<<?eppcar a :Permitted>> a :false. ?eppcar a soa:Pay;
                 soa:has-agent ?a; soa:has-instrument soa:cash}}"""]
```

After adding this prohibition, it is easy to see that a violation is inferred: from the RDF triples in (96), the SPARQL* rule in (98) generates and infers a new prohibition, namely that John is prohibited from paying in cash (in general, not only in Sketty):

(99)
```
<<_:b0 a :Permitted >> a :false.
_:b0 a soa:Pay; soa:has-agent soa:John; soa:has-instrument soa:cash.
```

However, since John *does* pay in cash, as specified in (96) and (97), the second rule shown above in (74) infers the following violation: the fact that John is prohibited to pay in cash is violated by the fact that he pays in cash at the parking meter in Sketty.

(100)
```
<< <<_b0 a :Permitted>> a :false>> :is-violated-by
<< <<epsj a :Rexist>> a :true>>
```

It is easy to see that, in (96), if the modality of `epsj` is changed from `Rexist` to `Obligatory`, a conflict is inferred rather than a violation: John would be obligated to pay in cash while simultaneously being prohibited from doing so.

Nevertheless, regardless of the modality of `epsj`, it is clear that the prohibition against paying in cash is incompatible with the necessity of doing so in Sketty. This incompatibility represents an additional form of "abnormality" that LegalTech applications should be able to automatically identify and report, alongside violations and conflicts. Note that, in the example under examination, this would also allow John to *appeal* the potential sanction for his violation by arguing that he had no choice but to violate the norm.

In order to do so, as already indicated at the beginning of this section, a new RDF property `is-necessarily-violated-by` is added to the vocabulary of the proposed computational ontology, along with the SPARQL* rule in (101). This rule states that if it is necessary for the thematic role of a given eventuality to have a specific value, but there is another more abstract eventuality that features that value in the same thematic role and is prohibited, then the fact that the latter is prohibited `is-necessarily-violated-by` the fact that it is necessary for that thematic role to have that value.

```
(101) [a :InferenceRule; :has-sparql-code """
        CONSTRUCT{<< <<?ep a :Permitted>> a :false>>
                  :is-necessarily-violated-by
                  << <<?en ?tr ?v>> a :necessary>>}
        WHERE{?tr a :ThematicRole. ?ep ?tr ?v. <<?ep a :Permitted>> a :false.
              ?en :instantiates ?ep. <<?en ?tr ?v>> a :necessary}"""]
```

The SPARQL* rule in (101) adds the following RDF triples to (100): the fact that John is prohibited to pay in cash `is-necessarily-violated-by` by the fact that it is necessary for the instrument of `soa:epsj` to be cash.

```
(102) << <<_b0 a :Permitted>> a :false>> :is-necessarily-violated-by
      <<soa:epsj soa:has-instrument soa:cash>> a :necessary
```

A symmetric SPARQL rule may be added to infer that, whenever it is *not possible* for a certain thematic role to take on a certain value, although this value is *obligatory* for a given eventuality, then the fact that the latter is obligatory `is-necessarily-violated-by` the fact that it is not possible for that thematic role to assume the specified value. Similarly, we may introduce a new RDF property `is-nullified-by` along with a SPARQL rule stating that, whenever it is *not possible* for a certain thematic role to take a given value, yet this value is *permitted* for a certain eventuality, then the permission `is-nullified-by` the fact that it is actually impossible to do what is permitted.

Developing an exhaustive list of properties that explicitly represent abnormalities between deontic statements and contextual constraints goes far beyond the scope of this paper. Therefore, the computational ontology available on GitHub does not include these additional properties or the corresponding SPARQL* rules.

8 Future works: resolvable conflicts and temporal management

To the best of our knowledge, this paper represents the first substantial attempt to unify contributions from three research strands, previously investigated largely in isolation, within a single framework: (1) RDF-based LegalTech solutions, (2) Natural Language Semantics via reification, and (3) conflict-tolerant approaches in deontic logic.

In addition, this paper addresses the incorporation of irresolvable conflicts into deontic logic. This is a topic of paramount importance in AI, as shown by extensive research in defeasible logic and argumentation theory over the past decades. However, it has received little attention within deontic logic itself. A notable exception is (Goble, 2013), to which this paper offers an alternative approach grounded in the legal theory of Hans Kelsen, one of the most influential jurists and legal philosophers of the 20th century.

Precisely because of the innovative and exploratory nature of the research presented here, much future work remains to be done.

Providing an exhaustive list of possible directions for future research would be overly extensive and not particularly useful. Therefore, this section only briefly discusses the two main aspects needed to make the proposed computational ontology suitable for deployment in real-world applications, which we identify as our next contributions to this topic: extending the ontology to address (1) *resolvable* conflicts and (2) temporal management.

8.1 Resolvable conflict

Identifying and notifying irresolvable conflicts and other abnormalities in legislation is of paramount importance. For this reason, the proposed solution provides a way to *explicitly* represent such conflicts, rather than merely associating conflicting deontic statements with consistent formulae that are indistinguishable from other consistent formulae.

However, once these conflicts are detected, it becomes the role of the legislator to resolve them. Since it is clearly impractical to rewrite the normative system from scratch each time a conflict is identified, the standard solution is to indicate within the normative code which of the two conflicting norms is stronger than the other one in the specific context where the conflict occurs, perhaps under specific interpretations of the legal concept occurring therein (cf. (Bartolini et al., 2016)).

Not surprisingly, norms from existing legislation often include numerous exceptions, sometimes even exceptions of exceptions, and so on. A common way to resolve conflicts when laws are amended is to add an *exception* to one of the conflicting norms, specifically to the one considered "weaker". In the particular context identified, the "weaker" norm does not apply because the "stronger" conflicting norm takes precedence.

In light of this, LegalTech applications require a formal mechanism to represent and process this conflict resolution strategy. (Robaldo et al., 2023) offers a recent comprehensive overview of the main computational approaches for implementing defeasibility and managing exceptions in normative reasoning. We plan to develop and integrate such a mechanism into the proposed computational ontology as well.

In (Robaldo et al., 2023) it is explained that two main alternative constructs have been used in formalizations of defeasible rules: *negation-as-failure* and *superiority relations*. The former is commonly used in reasoners based SHACL-SPARQL (Robaldo et al., 2023) and Answer Set Programming (ASP) (Calimeri et al., 2020), while the latter is found in systems such as DLV (Leone et al., 2006), PROLEG (Satoh et al., 2011), Arg2P (Billi et al., 2021), and SPINdle (Lam and Governatori, 2009).

According to (Satoh et al., 2011), superiority relations are generally more intuitive and easier to manipulate than negation-as-failure, particularly for legal practitioners.

The original version of PROLEG employed negation-as-failure, but based on experiments involving lawyers and law students, its developers later replaced it with an `exception` predicate that serves a similar role to superiority relations.

It is argued in (Robaldo et al., 2023) that the formalization of use cases in various formats supports Satoh et al.'s claims. It is simpler to explicitly state which rules override others through superiority relations than to encode overrides implicitly via special predicates relying on negation-as-failure. For example, managing defeasibility via negation-as-failure required multiple additional special predicates and classes in the ontology, complicating editing and revision. More complex cases would demand even more such predicates, which would be challenging for legal experts to track.

In contrast, a single meta-predicate such as PROLEG's `exception` can represent superiority relations. This operator allows for an intuitive definition of a directed acyclic graph indicating which rules override which. This simplicity makes superiority relations more accessible to lawyers and easier to maintain.

However, a key technical challenge remains. PROLEG is goal-oriented and evaluates rules top-down, allowing it to directly handle exceptions by attempting to satisfy exception goals before considering others. In contrast, SPARQL* rules and other reasoners based on ASP, though significantly more efficient than PROLEG, operate as forward-chaining production systems that generate inferences in a bottom-up manner.

Emulating PROLEG's goal-oriented exception handling within these frameworks is not straightforward. Therefore, our future research will focus on implementing superiority relations in a way that is compatible with forward-chaining reasoning. The work on SPINdle, which uses superiority relations and also operates as a forward-chaining reasoner, offers valuable inspiration in this regard. A notable limitation of SPINdle compared to SPARQL* and ASP is its inability to process RDF data directly, and its generally lower efficiency. However, this performance limitation has been mitigated in (Governatori, 2024), which presents an ASP-based implementation of SPINdle.

In summary, while negation-as-failure has proven to be a practical solution in existing systems, superiority relations provide a more intuitive and lawyer-friendly alternative. Our computational ontology aims to build on this insight while preserving efficiency. To this end, we plan to investigate future approaches for reconciling superiority relations with the proposed forward-chaining rules in SPARQL*.

8.2 Temporal management

All (but one) examples considered above in the paper are assumed to take place "here" and "now". In other words, the present paper does not consider the *temporal dimension*. Under the assumption that all eventualities occur "here" and "now", and that the same agent cannot perform the same action twice, it was possible to include the pragmatic implicature implemented in SPARQL* as shown in (16) above: if two eventualities `ei1` and `ei2` share the same type, modality, and agent, then the state of affairs includes an additional abstract eventuality instantiated by both. However, it is clear that the rule in (16) cannot be used as it is in contexts where eventualities are linked to specific time instants or intervals, for example through a thematic role `has-time`. In such cases, a more refined version of the rule is required.

The temporal dimension, together with the definition of constraints and inference rules for the `has-time` thematic role, will be addressed in our future work, along with the handling of resolvable conflicts discussed in the previous subsection.

It is well known that temporal information is essential for automated compliance checking and legal reasoning in general. This is not only because it helps identify the contextually relevant sets of bearers to which deontic statements apply, but also because every deontic statement is, either explicitly or implicitly, associated with a temporal validity and with temporal constraints within which obligations must be fulfilled.

This is especially true for deontic statements derived from existing legislation, which is the main focus of our research. Every legislative act is associated with a date on which it enters into force. The norms contained in the act may later be amended, with each amendment also carrying a date of effect. Nevertheless, the previous version of a norm continues to apply to facts that occurred in the time interval between the two dates.

In light of this, it is evident that the proposed computational ontology cannot yet be used in LegalTech applications for checking compliance with existing legislation until temporal management mechanisms are incorporated. While all examples discussed above involve eventualities taking place at the instant "now", additional SPARQL rules must be introduced to process deontic statements that refer to specific temporal *intervals*. For instance, consider the following sentences:

(103)
a. It is prohibited to enter the park from 3pm until 5pm.

b. John was in the park from 4pm until 6pm.

From (103.a-b), it can be inferred that John violated the obligation in (103.a) *only during the interval from 4pm to 5pm.* To enable this inference within the proposed computational ontology, it is necessary to introduce rules that identify the *overlap* between the two time intervals in (103.a-b) and construct a *new* interval representing their intersection. One possible solution is to add SPARQL rules that encode the well-known Allen's relations (Allen, 1984), and use the `CONSTRUCT` clause to generate the relevant new intervals in which the deontic statements are violated, complied with, or in conflict with one another.

Similar considerations apply to the intervals between the instant an obligation enters into force and the time at which the actual eventualities complying with that obligation take place. For example, if John receives a fine for not paying for parking, he will have a specific amount of time, say one month, to pay the fine. Otherwise, he will receive another fine. In the literature, the payment of a fine is usually represented as a further obligation (see (Governatori and Rotolo, 2019)): if John violates his obligation to pay for parking, he is then *obliged* to pay a fine within a certain period. The obligation to pay the fine replaces the obligation to pay for parking. If John pays the fine within the specified interval, the obligation is considered compensated by the payment. Otherwise, a new obligation to pay an increased fine replaces the previous one. Based on these considerations, (Governatori, 2015) presents an approach in which obligations are subcategorized according to the time at which the corresponding eventualities occur, and how the interval during which obligations are in force evolves after they are fulfilled.

In our future work, we plan to incorporate this subcategorization into the proposed computational ontology, using the Time Ontology[24] to represent time. The Time Ontology provides RDF resources for representing instants, intervals, and Allen's relations between intervals. We will introduce specific thematic roles to associate eventualities with instants or intervals, along with SPARQL* rules to perform the necessary inferences over temporal data. Since the Time Ontology is based on OWL, which does not support temporal reasoning constructs, it does not natively enable such inferences. This limitation makes it necessary to implement additional SPARQL* rules.

9 Conclusions

This paper presented a novel computational ontology for conflict-tolerant deontic reasoning, entirely implemented in RDF* and SPARQL*. To the best of our knowledge, this is the first substantial attempt to devise a deontic logic fully compatible with the Semantic Web, by integrating contributions from three research strands that have so far been investigated largely in isolation: (1) RDF-based LegalTech solutions, (2) Natural Language Semantics via reification, and (3) conflict-tolerant approaches in deontic logic.

A central contribution of this work is the explicit incorporation of *irresolvable conflicts* into a deontic logic framework. Although the importance of conflicts has been widely recognized in AI subfields such as defeasible reasoning and argumentation theory, they have rarely been addressed within deontic logic itself. A notable exception is (Goble, 2013), which this paper critically engages. In contrast to Goble's definition, which centers on an agent's inability to fulfill multiple obligations simultaneously, our approach follows Hans Kelsen's legal theory and is grounded in the concept of *violation*. A conflict, in this view, arises when complying with one norm or acting in accordance with what is permitted necessarily results in the violation of another norm, regardless of the agent's practical capabilities. This reconceptualization supports a more principled and legally grounded form of deontic reasoning.

In addition, what fundamentally distinguishes the proposed approach from the other conflict-tolerant deontic logics proposed in the literature is its capacity to *explicitly* represent abnormalities, such as contradictions, conflicts, and violations. This is a key departure from traditional logical systems, where such inconsistencies are treated as fatal errors. Standard reasoners typically *halt* execution upon encountering a contradiction. In contrast, our ontology supports *reasoning with* such abnormalities. This enables advanced reasoning tasks, such as identifying database inconsistencies, analyzing conflicting legal norms, or evaluating alternative law revisions.

Similar considerations were raised in a recent interview with Leon van der Torre and Dov Gabbay published in (Steen and Benzmuller, 2024), where the explicit representation of fallacies, violations, mistakes, and other abnormalities, enabling reasoning about them, was identified as a crucial gap in contemporary logical frameworks for AI.

Therefore, in the LegalTech application envisioned above, explicitly representing conflicts would not only allow the system to notify legislators about them, but also to reason about potential solutions, weigh the pros and cons of each, and thereby assist legislators in making better-informed decisions on how to revise the law.

[24]`https://www.w3.org/TR/owl-time`

This capability is made possible through the technique of *semantical embedding*, whereby elements typically confined to the proof theory, such as contradictions, are incorporated directly into the syntax of the logic. For example, the RDF property `is-in-contradiction-with` corresponds to the standard logical symbol $\perp$, which denotes a contradiction. Unlike $\perp$, which exists only in the meta-language used to formalize the logic's proof theory, `is-in-contradiction-with` is part of the object-level vocabulary, enabling contradictions to be represented, queried, and reasoned about directly with SPARQL*. This embedding of the proof theory into the logic aligns with methodologies developed in modern higher-order logic frameworks like LogiKEy (Benzmüller, Parent, and van der Torre, 2020), (Pasetto and Benzmüller, 2024), although, to the best of our knowledge, this is the first time such an approach has been implemented natively in RDF. This ensures full compatibility with W3C standards, making the framework suitable for integration within the broader Semantic Web ecosystem.

Another important contribution is the formalization of *all* deontic modalities outlined in the Deontic Traditional Scheme: obligations, permissions, optionality, and their negations, as well as various forms of violations and conflicts. Previous RDF-based approaches in this direction have typically been limited in scope and reliant on non-standard languages such as SWRL or constrained by the limited expressivity of OWL. By contrast, the proposed ontology employs SPARQL rules capable of expressing nuanced deontic interactions, including those dependent on contextual constraints and thematic roles.

Another novelty of our approach is the prominent inferential role assigned to the *permission* deontic modality. Unlike most traditional deontic logic frameworks, including Standard Deontic Logic (von Wright, 1951), which treat obligation as the foundational deontic modality, we consider permission to be the most informative deontic modality for detecting conflicts. As demonstrated by the SPARQL* rule shown above in (83), a conflict is inferred when two eventualities exist such that one is permitted, the other is not, and the permitted one is more specific than (instantiates) the other one. While an obligation entails permission, a permission does not entail an obligation but rather a *not*-obligation, which prevents direct conflict detection. Thus, basing the inferential framework on permissions rather than obligations allows for a more general and effective treatment of conflicts among deontic statements.

These reflections culminate in a novel framework that *significantly generalizes* the one proposed in (Goble, 2013), which, as discussed above, is widely regarded as the state of the art for representing normative conflicts in deontic logic. Unlike (Goble, 2013), which focuses solely on obligations and operates at the level of propositional formulas, our framework incorporates a broader range of deontic modalities and contextual constraints. Most importantly, as explained in Section 7, the definition of normative conflict adopted in (Goble, 2013), formalized as $\mathtt{OB(A)} \wedge \mathtt{OB(B)} \wedge \neg\Diamond(\mathtt{A} \wedge \mathtt{B})$, is understood within our framework as a special case of the interaction between obligations and contextual constraints. In our model, this corresponds to the formula $\mathtt{OB(C)} \wedge \neg\Diamond(\mathtt{C})$, of which Goble's definition constitutes a special case: where $\mathtt{C}$ is logically equivalent to $\mathtt{A} \wedge \mathtt{B}$.

While the work presented here is foundational, it has broad and significant implications. The formalization of norms is not merely a theoretical exercise; it is essential for building reliable AI systems in areas such as eHealth compliance, financial regulation, and other data-intensive domains where normative constraints are deeply embedded. Legal norms serve not only as restrictions on data but also as instruments for structuring and enriching it. Matching and annotating Big Data with legislative information can generate even larger and more valuable datasets. In this sense, research in LegalTech and compliance checking is key to advancing a wide range of disciplines and industrial applications, especially since effective legal systems are a critical factor in economic development.

Given the widespread adoption of Semantic Web technologies across both the public and private sectors, we consider the development of deontic logics fully compatible with RDF to be essential. In our view, this is the only way to ensure the interoperability, scalability, and practical integration of normative reasoning into real-world applications.

References

[Allen1984] Allen, J.F. 1984. Towards a general theory of action and time. *Artificial Intelligence*, 23(2).

[Anim, Robaldo, and Wyner2024] Anim, J., L. Robaldo, and A. Wyner. 2024. A SHACL-Based Approach for Enhancing Automated Compliance Checking with RDF Data. *Information*, 15.

[Bartolini et al.2016] Bartolini, Cesare, Andra Giurgiu, Gabriele Lenzini, and Livio Robaldo. 2016. Towards legal compliance by correlating standards and laws with a semi-automated methodology. In *BNCAI*, volume 765 of *Communications in Computer and Information Science*, pages 47–62. Springer.

[Bellomarini et al.2022] Bellomarini, L., D. Benedetto, G. Gottlob, and E. Sallinger. 2022. Vadalog: A modern architecture for automated reasoning with large knowledge graphs. *Information Systems*, 105.

[Benzmüller, Parent, and van der Torre2020] Benzmüller, C., X. Parent, and L. van der Torre. 2020. Designing normative theories for ethical and legal reasoning: Logikey framework, methodology, and tool support. *Artificial Intelligence*, 287.

[Billi et al.2021] Billi, Marco, Roberta Calegari, Giuseppe Contissa, Francesca Lagioia, Giuseppe Pisano, Galileo Sartor, and Giovanni Sartor. 2021. Argumentation and defeasible reasoning in the law. *J, special issue "The Impact of Artificial Intelligence on Law"*, 4(4).

[Calimeri et al.2020] Calimeri, F., C. Dodaro, D. Fuscà, S. Perri, and J. Zangari. 2020. Efficiently coupling the I-DLV grounder with ASP solvers. *Theory and Practice of Logic Programming*, 20(2).

[Ceci2013] Ceci, M. 2013. Representing judicial argumentation in the semantic web. In P. Casanovas, P. Pagallo, M. Palmirani, and G. Sartor, editors, *AI Approaches to the Complexity of Legal Systems*, volume 8929 of *Lecture Notes in Computer Science*. Springer.

[Davidson1967] Davidson, D. 1967. The logical form of action sentences. In Nicholas Rescher, editor, *The Logic of Decision and Action*. University of Pittsburgh Press.

[De Vos et al.2019] De Vos, Marina, Sabrina Kirrane, Julian A. Padget, and Ken Satoh. 2019. ODRL policy modelling and compliance checking. In Paul Fodor, Marco Montali, Diego Calvanese, and Dumitru Roman, editors, *Rules and Reasoning - Third International Joint Conference, RuleML+RR*.

[Faroldi and van De Putte2023] Faroldi, F. and F. van De Putte, editors. 2023. *Kit Fine on Truthmakers, Relevance, and Non-Classical Logic*. Springer Verlag.

[Francesconi and Governatori2023] Francesconi, E. and G. Governatori. 2023. Patterns for legal compliance checking in a decidable framework of linked open data. *Artificial Intelligence and Law*, 53(3).

[Frijters, Meheus, and van De Putte2021] Frijters, S., J. Meheus, and F. van De Putte. 2021. Reasoning with rules and rights : term-modal deontic logic. In Rahman, S. and Armgardt, M. and Kvernenes, H., editor, *New developments in legal reasoning and logic : from ancient law to modern legal systems*, volume 23 of *Logic, Argumentation & Reasoning*. Springer.

[Gabbay et al.2013] Gabbay, D., J. Horty, X. Parent, R. van der Meyden, and L. (eds.) van der Torre. 2013. *Handbook of Deontic Logic and Normative Systems*. College Publications.

[Gandon, Governatori, and Villata2017] Gandon, F., G. Governatori, and S. Villata. 2017. Normative requirements as linked data. In A. Wyner and G. Casini, editors, *Legal Knowledge and Information Systems.*, volume 302. IOS Press.

[Gentzen1964] Gentzen, G. 1964. Investigations into logical deduction. *American Philosophical Quarterly*, 1(4).

[Glimm et al.2014] Glimm, B., I. Horrocks, B. Motik, G. Stoilos, and Z. Wang. 2014. HermiT: An OWL 2 Reasoner. *Journal of Automated Reasoning*, 53(3).

[Goble2013] Goble, L. 2013. Prima facie norms, normative conflicts, and dilemmas. In D. Gabbay, J. Horty, X. Parent, R. van der Meyden, and L. van der Torre, editors, *Handbook of Deontic Logic and Normative Systems*. College Publications.

[Goble2014] Goble, L. 2014. Deontic logic (adapted) for normative conflicts. *Logic Journal of the IGPL*, 22(2).

[Gordon and Hobbs2017] Gordon, A.S. and J.R. Hobbs. 2017. *A formal theory of commonsense psychology - how people think people think*. Cambridge University Press.

[Gordon2008] Gordon, T. 2008. Constructing legal arguments with rules in the legal knowledge interchange format (LKIF). In Pompeu Casanovas, Giovanni Sartor, Núria Casellas, and Rossella Rubino, editors, *Computable Models of the Law*. Springer.

[Governatori2015] Governatori, G. 2015. The regorous approach to process compliance. In J. Kolb, B. Weber, S. Hallé, W. Mayer, A. K. Ghose, and G. Grossmann, editors, *19th IEEE International Enterprise Distributed Object Computing Workshop, EDOC Workshops*. IEEE Computer Society.

[Governatori2024] Governatori, G. 2024. An ASP implementation of defeasible deontic logic. *Künstliche Intelligence*, 38(1-2).

[Governatori and Rotolo2019] Governatori, Guido and Antonino Rotolo. 2019. Time and compensation mechanisms in checking legal compliance. *Journal of Applied Logics - IfCoLog Journal*, 6(5):815–846.

[Hansen2008] Hansen, J. 2008. Prioritized conditional imperatives: problems and a new proposal. *Autonomous Agents and Multi-Agent Systems*, 17(1).

[Hobbs2003] Hobbs, J.R. 2003. The logical notation: Ontological promiscuity. In *Chapter 2 of Discourse and Inference*. Available at http://www.isi.edu/∼hobbs/disinf-tc.html.

[Hoekstra et al.2007] Hoekstra, R., J. Breuker, M. Di Bello, and A. Boer. 2007. The LKIF core ontology of basic legal concepts. In P. Casanovas, M. Biasiotti, E. Francesconi, and M. Sagri, editors, *Proc. of the Workshop on Legal Ontologies and Artificial Intelligence Techniques (LOAIT 2007)*.

[Horty1994] Horty, J. 1994. Moral dilemmas and nonmonotonic logic. *Journal of Philosophical Logic*, 23.

[Horty2012] Horty, J.F. 2012. *Reasons as Defaults*. Oxford University Press, USA.

[Jago2020] Jago, Mark. 2020. Truthmaker semantics for relevant logic. *Journal of philosophical logic*, 49(4):681–702.

[Kelsen1991] Kelsen, H. 1991. Conflicts of Norms. In *General Theory of Norms*. Oxford University Press.

[Knuutila1981] Knuutila, S. 1981. The emergence of deontic logic in the fourteenth century. In R. Hilpinen, editor, *New Studies in Deontic Logic*. Dordrecht.

[Lam and Governatori2009] Lam, Ho-Pun and Guido Governatori. 2009. The Making of SPINdle. In Adrian Paschke, Guido Governatori, and John Hall, editors, *Proc. of International Symposium on Rule Interchange and Applications (RuleML 2009)*, Available online at `http://spindle.data61.csiro.au/spindle`. Springer-Verlag.

[Leone et al.2006] Leone, Nicola, Gerald Pfeifer, Wolfgang Faber, Thomas Eiter, Georg Gottlob, Simona Perri, and Francesco Scarcello. 2006. The dlv system for knowledge representation and reasoning. *ACM Transactions on Computational Logic*, 7(3).

[Makinson and van der Torre2000] Makinson, David and Leendert W. N. van der Torre. 2000. Input/output logics. *Journal of Philosophical Logic*, 29(4):383–408.

[McNamara1996a] McNamara, P. 1996a. Doing well enough: Toward a logic for commonsense morality. *Studia Logica*, 57(1).

[McNamara1996b] McNamara, P. 1996b. Making room for going beyond the call. *Mind*, 105(419).

[Palmirani and Governatori2018] Palmirani, M. and G. Governatori. 2018. Modelling legal knowledge for GDPR compliance checking. In M. Palmirani, editor, *Legal Knowledge and Information Systems - JURIX*, volume 313 of *Frontiers in Artificial Intelligence and Applications*, pages 101–110. IOS Press.

[Parent and van der Torre2014] Parent, X. and L. van der Torre. 2014. "Sing and Dance!" - Input/Output Logics without Weakening. In *Proc. of 12th International Conference in Deontic Logic and Normative Systems (DEON 2014)*.

[Parry1989] Parry, W. 1989. Analytic implication; its history, justification and varietiess. In J. Norman and R. Sylvan, editors, *Directions in Relevant Logic*. Springer Netherlands, Dordrecht.

[Pasetto and Benzmüller2024] Pasetto, L. and C. Benzmüller. 2024. Implementing the fatio protocol for multi-agent argumentation in logikey. In C. Benzmüller, J. Otten, and R. Ramanayake, editors, *Proc. of the 5th International Workshop on Automated Reasoning in Quantified Non-Classical Logics (ARQNL 2024)*, volume 3875 of *CEUR Workshop Proceedings*. CEUR-WS.org.

[Perez, Arenas, and Gutierrez2006] Perez, Jorge, Marcelo Arenas, and Claudio Gutierrez. 2006. Semantics and complexity of sparql.

[Robaldo2021] Robaldo, L. 2021. Towards compliance checking in reified I/O logic via SHACL. In Juliano Maranhão and Adam Zachary Wyner, editors, *Proc. of 18th International Conference for Artificial Intelligence and Law (ICAIL 2021)*. ACM.

[Robaldo et al.2020] Robaldo, L., C. Bartolini, M. Palmirani, A. Rossi, M. Martoni, and G. Lenzini. 2020. Formalizing GDPR provisions in reified I/O logic: the DAPRECO knowledge base. *The Journal of Logic, Language, and Information*, 29.

[Robaldo et al.2023] Robaldo, L., S. Batsakis, R. Calegari, F. Calimeri, M. Fujita, G. Governatori, M. Morelli, F. Pacenza, G. Pisano, K. Satoh, I. Tachmazidis, and J. Zangari. 2023. Compliance checking on first-order knowledge with conflicting and compensatory norms - a comparison among currently available technologies. *Artificial Intelligence and Law (to appear)*.

[Robaldo and Liga2025] Robaldo, L. and D. Liga. 2025. On the interplay between entailments among obligations and their violations. In Y. Nakano and T. Suzumura, editors, *New Frontiers in Artificial Intelligence*. Springer Nature Singapore.

[Robaldo et al.2023] Robaldo, L., F. Pacenza, J. Zangari, R. Calegari, F. Calimeri, and G. Siragusa. 2023. Efficient compliance checking of rdf data. *Journal of Logic and Computation (to appear)*.

[Robaldo and Sun2017] Robaldo, L. and X. Sun. 2017. Reified input/output logic: Combining input/output logic and reification to represent norms coming from existing legislation. *The Journal of Logic and Computation*, 7.

[Robles and Méndez2010] Robles, G. and J. M. Méndez. 2010. A Routley-Meyer type semantics for relevant logics including Br plus the Disjunctive Syllogism. *Journal of philosophical logic*, 39(2).

[Robles and Méndez2011] Robles, G. and J. M. Méndez. 2011. A Routley-Meyer semantics for relevant logics including TWR plus the Disjunctive Syllogism. *Logic Journal of IGPL*, 19(1).

[Routley and Meyer1972a] Routley, R. and R. Meyer. 1972a. The semantics of entailment II. *Journal of Philosophical Logic*, 1(1).

[Routley and Meyer1972b] Routley, R. and R. Meyer. 1972b. The semantics of entailment III. *Journal of Philosophical Logic*, 1(1).

[Routley and Meyer1973] Routley, R. and R. Meyer. 1973. The semantics of entailment I. In H. Leblanc, editor, *Truth, Syntax, and Semantics.* North-Holland.

[Satoh et al.2011] Satoh, Ken, Kento Asai, Takamune Kogawa, Masahiro Kubota, Megumi Nakamura, Yoshiaki Nishigai, Kei Shirakawa, and Chiaki Takano. 2011. PROLEG: An Implementation of the Presupposed Ultimate Fact Theory of Japanese Civil Code by PROLOG Technology. In Takashi Onada, Daisuke Bekki, and Elin McCready, editors, *New Frontiers in Artificial Intelligence*, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Searle1995] Searle, J. 1995. *The construction of social reality.* The Free Press, New York.

[Sergot2013] Sergot, M. 2013. Normative positions. In D. Gabbay, J. Horty, X. Parent, R. van der Meyden, and L. van der Torre, editors, *Handbook of Deontic Logic and Normative Systems.* College Publications.

[Steen and Benzmuller2024] Steen, A. and C. Benzmuller. 2024. What are Non-classical Logics and Why Do We Need Them? An Extended Interview with Dov Gabbay and Leon van der Torre. *Künstliche Intelligenz*, To appear.

[Sun and Robaldo2017] Sun, X. and L. Robaldo. 2017. On the complexity of input/output logic. *The Journal of Applied Logic*, 25:69–88.

[Sun and van der Torre2014] Sun, X. and L. van der Torre. 2014. Combining Constitutive and Regulative Norms in Input/Output Logic. In *Proc. of 12th International Conference in Deontic Logic and Normative Systems (DEON 2014).*

[van De Putte, Beirlaen, and Meheus2019] van De Putte, F., M. Beirlaen, and J. Meheus. 2019. Adaptive deontic logics: a survey. *Journal of Applied Logics - IfCoLog Journal*, 6(3).

[van der Torre and Tan2000] van der Torre, L. and Y.H. Tan. 2000. Two-phase deontic logic. *Logique et Analyse*, 43.

[von Wright1951] von Wright, G. 1951. Deontic logic. *Mind*, 60.

[Vranes2006] Vranes, E. 2006. The Definition of ‘Norm Conflict’ in International Law and Legal Theory. *European Journal of International Law*, 17(2).