On the interplay between entailments among obligations and their violations

Livio Robaldo[1], Davide Liga[2]

[1] School of Law, Swansea University, UK
livio.robaldo@swansea.ac.uk
[2] Department of Computer Science, University of Luxembourg
davide.liga@uni.lu

Abstract. In Standard Deontic Logic (SDL), it is established that if a statement `p` is obligatory, then any other statement `q` entailed by `p` is also obligatory. This principle is commonly referred to as `OB-RM`. However, the appropriateness of `OB-RM` in deontic and normative reasoning has been widely criticized, leading to the proposal of various (restricted) versions of the rule. This paper argues that `OB-RM` is also unsuitable when considering the *violations* associated with obligations, particularly the *penalties* that must be imposed when these violations occur. To address this issue, the paper introduces an extension of the RDF-based framework recently proposed in [20], which integrates violations, penalties, and entailments among obligations. The extended framework is available at https://github.com/liviorobaldo/jurisin2025.

Keywords: Deontic Logic · RDF & SPARQL · Violations & Penalties

1 Introduction

The starting point of this paper is the hypothesis that, in normative reasoning, obligations are always associated with *penalties* that must be applied in cases where the obligations are violated. If the law states specific obligations but then no penalty is applied to those who violate them, the law is clearly ineffective.

Often, the law precisely states the penalty associated with one or more obligations. For example, section 110(5) of the UK Equality Act 2010[3], reported in (1), states that those who violate any obligation from the same section will have to pay a monetary fine that depends on UK standard scale.

(1) A person guilty of an offence under subsection (4) is liable on summary conviction to a fine not exceeding level 5 on the standard scale.

Legislation does not always specify the penalty for violating a given obligation. In such cases, judges in courts or other designated authorities must determine an appropriate penalty for the violators. Moreover, penalties are not always monetary fines. Serious crimes are often punished with imprisonment. Conversely,

[3] https://www.legislation.gov.uk/ukpga/2010/15/section/110#section-110-5

minor offenses may result in non-monetary penalties; for example, if John is obligated to pay the rent for his apartment but fails to do so, he may be required to leave the apartment without necessarily incurring any financial penalty.

The association between obligations and penalties appears difficult to reconcile with the `OB-RM` rule from Standard Deontic Logic (SDL) [23]:

(2) If $\vdash \mathtt{p} \rightarrow \mathtt{q}$, then $\vdash \mathtt{OBp} \rightarrow \mathtt{OBq}$ (`OB-RM`)

The question is: if legislation specifies a penalty for the violation of `OBp`, what penalty applies to the violation of the more general obligation `OBq`? In fact, no penalty seems to be associated with `OBq`. Thus, `OB-RM` seems inadequate for normative reasoning, at least under the assumption stated at the beginning of the Introduction, i.e., that every obligation is always linked to a penalty.

On the other hand, `OB-RM` appears to be perfectly reasonable when considering the *truth-conditional meaning* of the propositions involved. This is likely why the rule was originally accepted in Standard Deontic Logic. To illustrate this, let us associate `p` and `q` with the natural language sentences in (3.a-b):

(3) a. `p`: John pays £100 in cash.
b. `q`: John pays £100.
c. `OBp`: John is obliged to pay £100 in cash.
d. `OBq`: John is obliged to pay £100.

If *it is true that* John pays £100 in cash, then *it is also true that* John pays £100. Then, `OB-RM` states that (3.c) entails (3.d): if *it is true that* John is obliged to pay £100 in cash, then *it is also true that* John is obliged to pay £100. While this is correct, it pertains to the truth-conditional meaning conveyed by material implication: if the antecedent is true, then the consequent must also be true.

Nevertheless, this truth-conditional perspective is orthogonal to the perspective considered here, where the obligations are viewed as a "list of tasks to be complied with by their bearers", who will incur penalties if they fail to do so.

This is not the first paper that proposes reasoning about obligations from a perspective different from standard truth-conditional semantics. In fact, the idea was originally introduced in [11], it is known as "the Jørgensen dilemma", and has been widely discussed in the deontic logic literature. The Jørgensen dilemma raises the question of how we can logically reason about norms, given that norms do not have truth values in the same way as descriptive statements. This paper argues that reasoning with norms involves reasoning about the violations and penalties associated with obligations, under the assumption that the normative system does not include obligations without penalties. If such obligations do exist, they are considered vacuous or ineffective, meaning that we can reason about them truth-conditionally, but not "deontically".

If these premises are accepted, lot of research in deontic logic still needs to be done. With the exception of Defeasible Deontic Logic (DDL) [6], most deontic logics proposed in the literature fail to properly formalize violations and penalties. Instead, they primarily introduce restricted variants of the `OB-RM` rule to mitigate its undesired effects.

This paper will extend the framework from [20] by incorporating constructs to link obligations with their penalties, along with rules to infer the latter from violations of the former. The objective is to accurately model the interaction between obligations, their entailments, and the penalties associated with their violations. The paper also compares the proposed approach with Defeasible Deontic Logic (DDL) and other deontic logics in the literature.

2 Related works

As explained in the Introduction above, to the best of our knowledge, all modern deontic logics but Defeasible Deontic Logic (DDL) [6] [5] do not represent/infer penalties associated with violations of obligations.

DDL achieves so by introducing the binary operator "$\otimes$" to relate two statements $\mathtt{p}$ and $\mathtt{q}$. The assertion $\mathtt{p} \otimes \mathtt{q}$ indicates[4] that $\mathtt{p}$ is obligatory, i.e., that $\mathtt{OBp}$ holds, and that if this obligation is violated (i.e., if $\neg\mathtt{p}$ also holds), then $\mathtt{q}$ is inferred as obligatory, i.e., $\mathtt{OBq}$ is inferred. In this way, $\mathtt{OBq}$ represents the penalty for violating $\mathtt{OBp}$, and is therefore modeled in DDL as another obligation.

For example, $\mathtt{OBq}$ could represent the obligation for the agent to pay a fine, or, as in the earlier example of John's rent mentioned in the Introduction, the obligation to vacate the apartment. Alternatively, $\mathtt{OBq}$ might be unspecified, meaning that it is left to a judge or another competent authority to determine the specific penalty for violators of $\mathtt{OBp}$. In such cases, it may be useful to assume that an obligation $\mathtt{p}$ for which the law does not specify a penalty is associated with a "functional" proposition $\mathtt{tbd(p)}$, where $\mathtt{tbd}$ stands for "to be decided" (by a judge or other competent authority). The $\mathtt{tbd}$ function can be easily formalized as a Skolem function that returns the penalty associated with the violation of its input proposition, which will be resolved contextually later (e.g., during a trial). For such cases, the DDL formula would then be expressed as $\mathtt{p} \otimes \mathtt{tbd(p)}$.

Thanks to the operator "$\otimes$", DDL enables a direct modeling of penalties. As noted earlier, to the best of our knowledge, DDL is unique in this regard: no other modern deontic logic offers such a clear representation of penalties.

Still, DDL seems problematic in light of the considerations above, as it uses a weaker version of $\mathtt{OB\text{-}RM}$ known as "rule conversion", formalized as follows [7]:

(4) If $\vdash \mathtt{a}_1, \mathtt{a}_2, \ldots, \mathtt{a}_n \rightarrow_{\mathtt{C}} \mathtt{b}$, then $\vdash \mathtt{OBa}_1, \mathtt{OBa}_2, \ldots, \mathtt{OBa}_n \rightarrow \mathtt{OBb}$

In (4), "$\rightarrow_{\mathtt{C}}$" is a restricted version of the standard implication operator "$\rightarrow$" from classical logic: "$\rightarrow_{\mathtt{C}}$" only allows for implications denoted by the constitutive rules occurring in the modelled normative code.

[4] The meaning of the operator $\otimes$ is actually more nuanced than this. For this reason, we use the verb "indicate" rather than the phrase "is equivalent to" in the main text. $\mathtt{OBq}$ represents a *compensatory* obligation, meaning that fulfilling $\mathtt{OBq}$ compensates for the violation of $\mathtt{OBp}$. In other words, $\mathtt{p} \otimes \mathtt{q}$ is *not* equivalent to $(\mathtt{OBp} \wedge \neg\mathtt{p}) \rightarrow \mathtt{OBq}$. While both expressions capture the penalty aspect, $\mathtt{p} \otimes \mathtt{q}$ also accounts for compensation. However, since compensations fall outside the scope of this paper, we assume for simplicity that the two expressions are equivalent.

The distinction between regulative rules and constitutive rules is well-known in the literature on normative reasoning. Regulative rules specify what is obligatory or permitted based on certain abstract concepts known as "institutional facts", whereas constitutive rules define what counts as institutional facts in the specific state of affairs where regulative rules are applied.

An archetypal example of constitutive rule, from the seminal work in [8], is whether "bicycles count as vehicles". Assuming they do, DDL's rule conversion stipulates that if John is obliged to buy a bicycle, he is also obliged to buy a vehicle. This seems incorrect from the perspective outlined here, as the question of what penalty John incurs for failing to buy a vehicle remains unanswered. Moreover, it is unclear whether and how DDL's rule conversion links the obligation derived through "$\otimes$" to a penalty (and, once again, *which* penalty?).

In contrast, we believe constitutive rules should not be used to expand the "list of tasks to be complied with by their bearers", as we called it earlier. Instead, they should be used to assess whether these tasks have been fulfilled. For example, if John is obliged to buy a vehicle and he buys a bicycle, his obligation is considered fulfilled in contexts where bicycles are counted as vehicles.

While DDL is the only modern deontic logic that offers a simple representation of penalties, other deontic logics incorporate restricted versions of `OB-RM`.

One such logic is the Input/Output (I/O) logic axiomatized in [16]. The original axiomatization of I/O logic, from [15], does include a derivation rule called "Weakening the output" (`WO`), which parallels `OB-RM`. [16] discusses how `WO`, like `OB-RM`, leads to undesirable inferences when dealing with certain deontic paradoxes [9] or conflicts between obligations [2]. In light of this, [16] proposes replacing `WO` with other derivation rules, one of which is the `EQ` rule. This rule infers an obligation `OBq` from another obligation `OBp` only when `q` and `p` are *equivalent*, i.e., only when $\mathtt{p} \leftrightarrow \mathtt{q}$ holds in the state of affairs.

A similar solution is proposed in [2], which introduces a deontic logic called "Basic Deontic Logic (BDL)". BDL is based on possible-world semantics, like SDL, but unlike SDL, it allows for conflicts between obligations[5]. In BDL, `OB-RM` is replaced by a stricter rule, called `RBE`:

(5) If $\vdash \mathtt{p} \leftrightarrow_{\mathtt{A}} \mathtt{q}$, then $\vdash \mathtt{OBp} \leftrightarrow \mathtt{OBq}$ (`RBE`)

"$\leftrightarrow_{\mathtt{A}}$" is a restricted version of the standard bi-implication "$\leftrightarrow$". The restriction is made to avoid the so-called "deontic explosion", which involves inferring that everything is obligatory from a conflict of obligations (see [2] for formal details).

However, neither I/O logic nor BDL includes an operator that parallels DDL's "$\otimes$". In other words, neither logic allows for the explicit representation and inference of the penalties associated with the violations of obligations.

Considering only propositions logically equivalent to the obligatory ones may be acceptable from the perspective of applying penalties, but only under the (reasonable) assumptions that:

[5] A conflict among obligations occurs when two or more obligations hold, but complying with one entails violating the others. SDL's axioms lead to an inconsistency in the face of conflicting obligations, while BDL's do not.

(6) a. Whenever obligations are either complied with or violated, then *also all their equivalent obligations are.*

b. The penalty for a set of equivalent obligations *is applied only once*, and not multiple times for each obligation in the set.

This paper proposes an extension of the RDF-based framework in [20] that aligns with these assumptions. Thus, the proposed solution lies somewhat between the approaches in I/O logic or BDL and that of DDL: like the former, it accepts only equivalent obligations; like the latter, it includes explicit constructs to represent penalties and rules to derive them from violations of the asserted obligations.

3 The RDF-based framework in [20]

The framework in [20] is a recent deontic logic implemented in RDF and SPARQL. As is well known, RDF is the W3C standard for representing data in the Semantic Web, while SPARQL is the W3C standard for querying and making inferences with RDF data. Given the rapid growth of Big Data on the World Wide Web, RDF is perhaps the most widely used format for knowledge representation. [20] aims at providing a logical framework for checking compliance on RDF data.

The proposal in [20] builds upon "reified Input/Output logic" [17], an extension of standard I/O logic designed to handle Natural Language Semantics. It has been demonstrated in [21] that norm-based semantics, such as the one adopted in (reified) I/O logic, is more computationally efficient than other deontic logics based on possible-world semantics. As a result, providing solutions that are fully interoperable with RDF is crucial, as the cost of converting RDF-encoded Big Data into the input format for other logical reasoners (e.g., SPINdle [13], the automated reasoner for DDL) would be prohibitively high. We refer the reader to [18] and [19] for an empirical comparison of the performance achieved by some of the main contemporary legal reasoners, including SPINdle, on RDF data.

Besides the practical advantages brought by RDF and SPARQL, [20] features several theoretical advantages over past proposals in deontic logic. It is a *conflict-tolerant* deontic logic, like DDL and the BDL logic mentioned earlier, meaning that it represents conflicts among obligations as consistent formulae. However, it encompasses a wider range of conflicts and better models the interplay between obligations and constraints holding in the state of affairs. The intuitions behind [20], inspired by the work of H. Kelsen [12] [22], stem from the alternative analysis that this paper is also employing: the meaning of obligations is investigated from the perspective of how obligations are either complied with or violated.

Last but not least, thanks to RDF *reification*[6], the framework in [20] is also capable of *explicitly* representing violations and other abnormal situations, namely to associate these with specific RDF individuals. Reification is also what will be used in this paper to link violations with the corresponding penalties, thus enabling the framework to infer the latter when the former occur.

[6] https://www.w3.org/TR/rdf11-mt/#reification

Let's now see some examples. Sentences (3.a) and (3.d) are respectively formalized, in Turtle format[7], as in (7.a-b). This paper assumes that the reader already has some basic knowledge of RDF and SPARQL formats.

(7) a. `soa:epj a :Rexist, soa:Pay; soa:has-agent soa:John; soa:has-object soa:100pounds; soa:has-instrument soa:cash.`

b. `soa:epoj a :Obligatory, soa:Pay; soa:has-agent soa:John; soa:has-object soa:100pounds.`

In (7.a-b), "`soa:`" is the RDF prefix associated with the resources in the state of affairs that we want to model, while the empty prefix "`:`" is the one for the RDF resources in [20]'s framework. Therefore, the state of affairs includes the action "Pay" (RDF class `soa:Pay`), its thematic roles "agent", "object", and "instrument" (properties `soa:has-agent`, `soa:has-object`, and `soa:has-instrument`), and the individuals John", £100", and cash" (individuals `soa:John`, `soa:100pounds`, and `soa:cash`). Finally, the individuals `soa:epj` and `soa:epoj` are called eventualities", from [3], and refer to the two actions of paying.

`:Rexist` and `:Obligatory` are two modalities defined in [20]. `:Rexist` stands for "really exist" and marks the eventualities that truly take place in the state of affairs. `:Obligatory`, on the other hand, marks the eventualities that are obligatory in the state of affairs (though they do not necessarily take place).

Thus, in (7.a), John does really pay £100 in cash while in (7.b) (7.b) John is *obliged* to pay £100, because the individual `soa:epoj` belongs to the class `:Obligatory`. On the other hand, in (7.b) it is unknown whether John also pays the £100, i.e., whether the eventuality also belongs to the class `:Rexist`.

In [20], RDF assertions can also be negated; however, the RDF triples that are negated are typically those where the object is one of the modalities, such as `:Rexist` and `:Obligatory`. Since RDF vocabulary does not include standard logical negation (usually represented by the symbol "¬"), [20] introduces two special RDF classes, `:false` and `:hold`, and reifies the triples as individuals of these classes. In RDF, a triple can be reified by creating a new individual in the class `rdf:Statement`, of which `:false` and `:hold` are subclasses, and linking this new individual to the three elements of the triple via the RDF properties `rdf:subject`, `rdf:predicate`, and `rdf:object`, respectively.

Therefore, John does not pay £100" is represented as in (8), where "`[...]`" refers to an anonymous RDF individual, also known as a blank node[8]. In (8), `soa:epoj` is the same eventuality defined in (7.b), which denotes the fact that John pays £100. (8) is the reification of the triple "`soa:epoj a :Rexist`", which is *not* part of the knowledge graph representing the state of affairs; in fact, (8) precisely states the opposite: this triple is false in the state of affairs. (8) corresponds to the standard first-order logic literal "¬`:Rexist(soa:epoj)`".

(8) `[a :false,:hold; rdf:subject soa:epoj; rdf:predicate rdf:type; rdf:object :Rexist].`

[7] https://www.w3.org/TR/turtle
[8] https://www.w3.org/TR/2014/REC-rdf11-mt-20140225/#blank-nodes

3.1 Abstract eventualities and their instantiations

Eventualities may feature different levels of abstraction, which is given by the set of specified versus non-specified thematic roles.

For example, in (7) the eventuality `soa:epoj` is *more abstract* than `soa:epj`. `soa:epoj` denotes the general/abstract fact "John pays £100"; however, John can do so in cash, by card, or with another instrument. Conversely, `soa:epj` denotes the fact "John pays £100 *in cash*", i.e., with that particular instrument.

In cases where an eventuality specifies a set of thematic roles that is a *superset* of the set of thematic roles of another eventuality, and the shared thematic roles have the same values, it is said that the former *instantiates* the latter.

The framework in [20] stipulates that an obligatory eventuality is *complied with* by any really existing eventuality that instantiates the obligatory one. Thus, in the example in (7), the fact that John is obliged to pay £100, i.e., `soa:epoj`, is complied with by the fact that John pays £100 in cash, i.e., `soa:epj`, but also by the fact that he pays them by card or with any other instrument: if any of these instantiations of `soa:epoj` really exists, then `soa:epoj` is complied with. This is enforced by the following SPARQL rule:

(9)
```
CONSTRUCT{[a :true,:hold; rdf:subject ?eo; rdf:predicate rdf:type;
      rdf:object :Obligatory] :is-complied-with-by [a :true,:hold;
      rdf:subject ?e;rdf:predicate rdf:type;rdf:object :Rexist]}
  WHERE{?eo a :Obligatory, ?c. ?e a :Rexist, ?c. ?c a :Eventuality.
   NOT EXISTS{?tr a :ThematicRole. ?eo ?tr ?vo. NOT EXISTS{?e ?tr ?ve}}
   NOT EXISTS{?tr a :ThematicRole. ?eo ?tr ?vo. ?e ?tr ?ve.
                                                FILTER(?vo!=?ve)}
   NOT EXISTS{[a :true,:hold; rdf:subject ?eo; rdf:predicate rdf:type;
      rdf:object :Obligatory] :is-complied-with-by [a :true,:hold;
      rdf:subject ?e;rdf:predicate rdf:type;rdf:object :Rexist]}}
```

The `WHERE` clause in (10) contains three `NOT EXISTS` sub-clauses. The first two ensure that the really existing eventuality `?e` indeed instantiates the obligatory eventuality `?eo`, as defined above, i.e., that the thematic roles in the former are a superset of those in the latter, and the shared roles have identical values. Specifically, these two sub-clauses verify that no thematic role is specified in `?eo` but missing in `?e`, and that no thematic role is specified in both `?eo` and `?e` with differing values (`?vo!=?ve`).

The third `NOT EXISTS` sub-clause is only included to prevent infinite loops, as detailed in [20], §2.1. In other words, this sub-clause serves purely *technical* purposes unrelated to the research presented in this paper. For space constraints, similar sub-clauses are omitted from the rest of the paper; readers can find the full version of the rules in the GitHub repository associated with this paper.

For each pair of eventualities (`?eo`, `?e`) for which the `WHERE` clause in (10) is true, the SPARQL rule constructs new RDF triples in the knowledge graph. Specifically, it asserts that the obligatory eventuality `?eo` is complied with by the really existing eventuality `?e` that instantiates `?eo`. The special RDF property `is-complied-with-by` explicitly connects obligations with their complying

facts. Since RDF properties only apply to RDF resources and not to RDF triples, the two triples "`?eo a :Obligatory`" and "`?e a :Rexist`" must be again reified. Thus, the content of the `CONSTRUCT` clause in (10) reads: "the fact that `?eo` is obligatory is complied with by the fact that `?e` really exists".

4 Representing violations and associated penalties

[20] defines a rule to infer when obligations are *complied with*, shown in (10). Conversely, it does not define rules to deduct when they are *violated*. [20] simply assumes that obligations are violated when they are not complied with.

The definition of specific inference rules needed to *deductively* infer violations of obligations is instead part of the original research presented in this paper.

In light of the distinction between abstract eventualities and their instantiations explained in the previous section, this paper stipulates that obligations are violated when (and only when) *all* their instantiations do not really exist. Thus, John violated his obligation of paying £100 only if it can be demonstrated that he did not pay them *in any way*, i.e., that he did not pay them in cash, nor by card, etc. Note that, given that RDF semantics follows the Open World Assumption, if the knowledge graph does not include the triple "`epj a :Rexist`", we cannot deduct that John did not pay £100: this is simply *unknown*.

In reality, it could be quite challenging to demonstrate whether certain (unknown) facts actually occurred. For instance, suppose that, in the example under consideration, the police needs to determine whether John made the payment. The police would have to show that he did not make any payment in cash, nor by card, nor through any other possible instrument with which John could have made the payment. Of course, this would be highly resource-intensive. For this reason, the bearers of obligations are often required to provide *evidence of compliance*. In the example at hand, John might also be obliged to request and keep an invoice for his payment, to present upon request. See related discussion in [4], where a similar solution has been proposed and formalized in DDL.

Still, it is sometimes possible to demonstrate, with a single inference, that the abstract eventuality did not really exist, by simultaneously proving the nonexistence of all its instantiations. For example, in the scenario under consideration, if it is established that John does not *own* any money, it can be inferred that he did not make any payment (whether £100, £200, or any other amount). This inference can be implemented within the framework of [20] using the following SPARQL rule, which specifies that if an individual does not own money, then s/he do not make a payment:

(10)
```
CONSTRUCT{[a :false,:hold;
            rdf:subject [a soa:Pay; soa:has-agent ?a];
            rdf:predicate rdf:type; rdf:object :Rexist]}
WHERE{?ro a :false,:hold; rdf:subject ?eo; rdf:predicate
      rdf:type; rdf:object :Rexist. ?eo a soa:Own;
      soa:has-agent ?a; soa:has-object soa:money}
```

Note that (10) creates *two* anonymous individuals. The first one represents the abstract eventuality corresponding to the act of paying performed by the agent `?a`, e.g., John, who does not own any money. The second one represents the reification of this abstract eventuality; this reification is necessary to assert that the abstract eventuality does not really exist.

Therefore, if the triples in (11.a) are asserted, those in (11.b) are inferred through (10): the (abstract) act of John's paying does not really exist.

(11) a.
```
soa:r a :false,:hold; rdf:subject soa:eojm; rdf:predicate
  rdf:type; rdf:object :Rexist. soa:eojm a soa:Own;
  soa:has-agent soa:John; soa:has-object soa:money.
```

b.
```
[a :false,:hold; rdf:subject [a soa:Pay; soa:has-agent
  soa:John]; rdf:predicate rdf:type; rdf:object :Rexist].
```

Since it does not really exist that John makes a payment (of any amount, with any instrument, etc.), as John does not own any money, the obligation in (7.b) cannot be complied with, i.e., it can be *deductively* inferred that it is violated.

This inference can be made using the SPARQL rule in (12), which is the dual of (10): while (10) searches for eventualities that really exist and instantiate the obligatory eventuality, (12) searches for eventualities that do not really exist and of which the obligatory one is an instantiation.

(12)
```
CONSTRUCT{[a :true,:hold; rdf:subject ?eo; rdf:predicate rdf:type;
      rdf:object :Obligatory] :is-violated-by ?r}
  WHERE{?eo a :Obligatory, ?c. ?e a ?c. ?c a :Eventuality.
   ?r a :false,:hold; rdf:subject ?e; rdf:predicate rdf:type;
                                          rdf:object :Rexist.
   NOT EXISTS{?tr a :ThematicRole. ?e ?tr ?vo. NOT EXISTS{?eo ?tr ?ve}}
   NOT EXISTS{?tr a :ThematicRole. ?e ?tr ?vo. ?eo ?tr ?ve.
                                            FILTER(?vo!=?ve)}}
```

From (7.b) and (11.b), (12) infers the triples in (13): the fact that John is obliged to pay £100 is violated by the fact that John does not make any payment.

(13)
```
[a :true,:hold; rdf:subject soa:epoj; rdf:predicate rdf:type;
  rdf:object :Obligatory] :is-violated-by [a :false,:hold;
  rdf:subject [a soa:Pay; soa:has-agent soa:John];
  rdf:predicate rdf:type; rdf:object :Rexist]
```

4.1 Representing penalties

After extending [20] to infer violations of obligations, this subsection further expands it to represent and infer the penalties associated with those violations.

As explained at the end of Section 2, this paper follows the approach in [6], which represents penalties as new obligations added to the knowledge graph whenever the corresponding violations are inferred.

In our running example, let us assume that, in (7.b), `epoj` denotes John's obligation to pay the rent to Bill. This can be encoded by adding appropriate RDF triples, such as "`soa:epoj soa:has-recipient soa:Bill`", which are omitted here for space reasons. Additionally, assume that if John fails to pay the rent, he is obliged to vacate Bill's apartment. This is encoded as follows:

(14)
```
soa:epoj :has-penalty soa:pepoj.

soa:pepoj a soa:Leave; soa:has-agent soa:John;
soa:has-object soa:BillApt.

[a :true; rdf:subject soa:pepoj; rdf:predicate rdf:type;
rdf:object :Obligatory].
```

In (14), `soa:pepoj` represents the penalty associated with `soa:epoj`, which is John's obligation to pay Bill's rent. The RDF property `has-penalty` corresponds to DDL's operator "$\otimes$". `soa:pepoj` denotes the act of leaving Bill's apartment, performed by John. Finally, the anonymous individual at the bottom of (14) represents the truth of the triple "`soa:pepoj a :Obligatory`". However, and this is crucial to understand, `soa:pepoj` *does not hold in the state of affairs*. For this reason, the anonymous individual at the bottom of (14) has been asserted only as an instance of the class `:true`, but *not* as an instance of the class `:hold`.

Therefore, John is not currently obliged to leave the apartment. He will only be required to do so if he violates the obligation `soa:epoj`. To enforce this, the SPARQL rule in (15) is added. For every violated obligation, this rule infers that the fact of its corresponding penalty being true *holds* in the state of affairs:

(15)
```
CONSTRUCT{?r2 a :hold}
WHERE{?r1 :is-violated-by ?e; a :true,:hold; rdf:subject ?eo;
      rdf:predicate rdf:type; rdf:object :Obligatory.
      ?eo :has-penalty ?p. ?r2 a :true; rdf:subject ?p;
      rdf:predicate rdf:type; rdf:object :Obligatory}
```

From (13) and (14), (15) infers the triples in (16.a). These are, in turn, inferred to be equivalent to the triple in (16.b) by one of the rules in [20][9], which we omit.

(16) a.
```
[a :true, :hold; rdf:subject soa:pepoj;
    rdf:predicate rdf:type; rdf:object :Obligatory].
```

b.
```
soa:pepoj a :Obligatory
```

As already discussed in the Introduction, legislation does not always associate obligations with specific penalties. Furthermore, since penalties are modeled as obligations, they might also be violated and must, therefore, be associated with another penalty in turn. For example, what happens if John is obliged to leave Bill's house because he did not pay the rent, but he does not?

[9] Specifically, by the SPARQL rule (28) in [20], §3.3.1.

In existing legislation, for norms not *explicitly* associated with penalties, it is *implicitly* assumed that a judge or another appointed authority will determine the appropriate penalty during a trial or alternative legal procedure. For example, in the John-Bill scenario, Bill might eventually call the police, who would force John to leave the house and possibly impose an additional penalty, such as a fine to cover the police's expenses.

To account for unspecified penalties, this paper introduces an additional RDF class "`TBD`", which stands for "to be decided" and parallels the function "`tbd`" introduced for DDL above in Section 2. Obligations with unknown penalties are linked to anonymous individuals in this class. These anonymous individuals can be regarded as Skolemizations of unspecified penalties, which are expected to be resolved at a later stage by an appointed authority.

Thus, the penalty `soa:pepoj` in (14) is associated with the "penalty on the penalty" `[a :TBD]` in (17). This represents what John will be obliged to do if he violates `soa:pepoj`. Therefore, in situations where `soa:pepoj` is obligatory but John does not leave Bill's house, the SPARQL rule in (15) will infer that the anonymous individual `[a :TBD]` becomes obligatory. Further examples are available in the GitHub repository associated with this paper.

(17)
```
soa:pepoj :has-penalty soa:ppepoj.

soa:ppepoj rdf:type :TBD.

[a :true; rdf:subject soa:ppepoj; rdf:predicate rdf:type;
rdf:object :Obligatory].
```

In the proposed solution, every obligation that is not associated with an explicit penalty is by default associated with an (unspecified) individual of the class `TBD`.

5 Entailments among obligations

Having incorporated in [20] an account of violations and associated penalties, this section addresses the initial research question: whether, and to what extent, entailments among obligations can be formalized.

Section 2 above provided arguments suggesting that the `OB-RM` rule of SDL does not align with the approach of associating every obligation with a penalty, which is instead endorsed in this paper. Specifically, it is unclear which penalties should be associated with the obligations entailed by `OB-RM`, which are truth-conditionally *more general* than those in the antecedent. It cannot be straightforwardly claimed that these penalties are merely unspecified, at least not in the sense used in the previous section. Instead, it appears that no appropriate penalty can be associated with these inferred obligations, which may justify the conclusion that such obligations should not be inferred at all.

Conversely, *equivalent* obligations seem to be acceptable from this perspective, at least under the two assumptions in (6) above: (1) if an obligation is either complied with or violated, then all its equivalent obligations are; (2) the penalty is applied only once for all obligations within a set of equivalent obligations.

In light of this, the solution proposed in this paper parallels the EQ derivation rule in I/O logic [16] or the RBE axiom[10] in BDL [2].

For instance, in the running example of this paper, let's assume that Bill also accepts euros as payment, possibly because he rents his apartment in the UK but lives in France, and thus finds it convenient to use euros in his daily life.

Given that £100 is equivalent to €118 (based on today's currency exchange rate), it can be assumed that the act of John paying £100, formalized as in (7.b), is equivalent to the act of John paying €118, formalized as in (18). The latter includes a new RDF property, is-equivalent-to, which explicitly encodes the equivalence between the two eventualities.

(18)
```
soa:epoj2 a soa:Pay; soa:has-agent soa:John;
soa:has-object soa:118euros; :is-equivalent-to soa:epoj.
```

Note that the equivalence between soa:epoj2 and soa:epoj holds in the present state of affairs, where Bill has agreed to also accept euros from John, but it does not necessarily hold *in general*. In other words, it cannot always be asserted that, in *any* state of affairs, paying £100 is equivalent to paying €118.

Thanks to the new RDF property is-equivalent-to, it is straightforward to implement SPARQL rules that parallel the rule EQ in [16] and the axiom RBE in [2], while also incorporating the assumptions in (6.a-b). These SPARQL rules are shown in (19) and (20).

Since in [20] every class that denotes a modality, e.g., Rexist and Obligatory, is asserted as an individual of the top class Modality, the SPARQL rule in (19) infers that the modality holding on an eventuality also holds on all eventualities reachable from the former through a chain of is-equivalent-to properties.

(19)
```
CONSTRUCT{?e2 a ?m}
WHERE{?m rdf:type/rdfs:subClassOf* :Modality. ?e1 a ?m.
      ?e1 (:is-equivalent-to|^:is-equivalent-to)+ ?e2}
```

On the other hand, the SPARQL rule in (20) infers that if the reification of an obligation is either complied with or violated by the reification of another eventuality, then the reifications of all equivalent obligations are.

(20)
```
CONSTRUCT{[a :true,:hold; rdf:subject ?e2;
      rdf:predicate rdf:type; rdf:object :Obligatory] ?cv ?r2}
WHERE{?r1 ?cv ?r2. ?r1 a :true,:hold; rdf:subject ?e1;
    rdf:predicate rdf:type; rdf:object :Obligatory.
    FILTER((?cv=:is-complied-with-by)||(?cv=:is-violated-by))
    ?e1 (:is-equivalent-to|^:is-equivalent-to)+ ?e2}
```

Therefore, on the one hand, if a really existing action complies with an obligation, the SPARQL rule in (20) infers that the same action complies with all its

[10] Contrary to BDL, this paper does not use a restricted meaning of bi-implication (e.g., from $\leftrightarrow$ to the $\leftrightarrow_A$ operator used in RBE), because [20] already blocks deontic explosion by rejecting the rule of Disjunctive Introduction (see [20] for details).

equivalent obligations. On the other hand, the same rule allows the SPARQL rule in (15) to still infer the penalties associated with obligations, even in cases where one of its equivalent obligation is violated.

For example, if John pays €118 to Bill, the rule shown above in (10) infers that `soa:epo2j` in (18) is complied with by this payment; subsequently, the rule in (20) infers that `soa:epoj` is also complied with.

Note that penalties are not *duplicated* for each equivalent obligation. Instead, it is the compliance/violation of the obligations that is *propagated* to all equivalent obligations. Indeed, if penalties were duplicated or propagated, the SPARQL rule in (15) would infer *multiple* penalties, one for each equivalent obligation, thereby contradicting what has been stipulated in (6.b).

6 Conclusions and Future Works

This paper extends the RDF-based framework recently proposed in [20] with RDF resources and SPARQL rules that parallel DDL's operator "⊗", in order to represent and infer violations of obligations and their corresponding penalties.

Nevertheless, contrary to DDL, this paper does not consider DDL's rule conversion as a valid inference rule, nor the more general rule `OB-RM` from SDL. The reason is that, in our view, these implications only concern the truth-conditional meaning of the natural language sentences conveying the obligations, whereas they should not add further obligations to the "list of tasks that must be complied with", as we informally referred to it in the Introduction.

On the other hand, this paper enables the representation of equivalent obligations, as it is done in the I/O logic proposed in [16] or the BDL logic proposed in [2], provided that: (1) whenever obligations are either complied with or violated, all their equivalent obligations are; and (2) the penalty for a set of equivalent obligations is applied only once, not multiple times for each obligation in the set. The RDF resources and SPARQL rules introduced in this paper represent and enable inferences on violations of obligations and corresponding penalties, while achieving (1) and (2).

In principle, solutions that parallel the RDF resources and SPARQL rules proposed here, or the operator "⊗" proposed in DDL, could also be implemented in I/O logic or BDL. However, implementing them in I/O logic or BDL seems more formally complex, as it would require representing and enabling inferences on *meta-assertions* about the obligations, which could be implemented by labeling the obligations and introducing additional rules that act on these labels.

Conversely, the high degree of expressivity offered by RDF, particularly through its reification mechanism, provides a more flexible formal account. Obligations can be reified into, i.e., associated with, individuals in the same format, thus integrating the meta-level within the same underlying level.

RDF is a highly expressive yet relatively easy-to-use knowledge representation format, while the alternative logical formats considered in past deontic logic literature do not seem to offer an equivalent expressivity-to-usability ratio.

Another key difference between the approach proposed here and the three approaches from the literature discussed in the "Related Works" section above is that the approach here stipulates that every obligation is *always* associated with a penalty (unless it is asserted as equivalent to another obligation already associated with that penalty). Furthermore, it could be the case that this penalty is "currently unknown", meaning it will need to be decided by a judge or another appointed authority during a trial or similar legal procedures.

In the approach proposed here, currently unknown penalties are represented as individuals of a special RDF class `TBD`, which stands for "to be decided". These individuals are Skolem constants that represent unspecified penalties and that may be later "resolved" into concrete penalties that violators will be obliged to fulfill. In this context, it could be highly beneficial, for risk assessment purposes, to develop a LegalTech application that associates individuals in the class `TBD` with *estimates* of the penalties they could represent, based on penalties assigned to violators of similar obligations in past case law. The solution proposed in this paper could therefore serve as the underlying framework for such an application, which might be considered as a potential direction for future work.

More generally, in our future work we aim to explore how the RDF representations presented here can be automatically extracted from legislative texts using Natural Language Processing (NLP) techniques [1] [10] [14]. The goal is to develop methods that allow for the automatic parsing of legal documents and the generation of RDF representations that capture the structure and semantics of the obligations and penalties described in the legislation. This would significantly streamline the process of creating structured legal data, reducing manual effort and increasing the efficiency of legal analysis. By integrating NLP with the RDF framework, we hope to make it easier to work with large volumes of legislation, enabling automated reasoning and more accurate predictions regarding the enforcement of legal obligations and penalties.

Acknowledgments

We are deeply grateful to Guido Governatori for our discussions on the preliminary formalizations shown in this paper, which greatly enriched the research presented. This research was supported by Innovate UK project 10106412: "Odyssey - Opening the National Archive's Legal Data to AI for Access to Justice (A2J)".

References

1. Boella, G., Di Caro, L., Rispoli, D., Robaldo, L.: A system for classifying multi-label text into eurovoc. In: Francesconi, E., Verheij, B. (eds.) Proc. of International Conference on Artificial Intelligence and Law, ICAIL. ACM (2013)
2. Goble, L.: Prima facie norms, normative conflicts, and dilemmas. In: Gabbay, D., Horty, J., Parent, X., van der Meyden, R., van der Torre, L. (eds.) Handbook of Deontic Logic and Normative Systems. College Publications (2013)
3. Gordon, A., Hobbs, J.: A formal theory of commonsense psychology - how people think people think. Cambridge University Press (2017)

4. Governatori, G.: Burden of compliance and burden of violation. In: Proc. of 28th Annual Conference in Legal Knowledge and Information Systems (JURIX 2015). Frontiers in Artificial Intelligence and Applications, vol. 279. IOS Press (2015)
5. Governatori, G.: An ASP implementation of defeasible deontic logic. Künstliche Intelligenz **38**(1) (2024)
6. Governatori, G., Olivieri, F., Rotolo, A., Scannapieco, S.: Computing strong and weak permissions in defeasible logic. Journal of Philosophical Logic **6**(42) (2013)
7. Governatori, G., Rotolo, A.: Deontic ambiguities in legal reasoning. In: Grabmair, M., Andrade, F., Novais, P. (eds.) Proc. of the 19th International Conference on Artificial Intelligence and Law, ICAIL. ACM (2023)
8. Grossi, D., Meyer, J., Dignum, F.: The many faces of counts-as: A formal analysis of constitutive rules. Journal of Applied Logic **6**(2) (2008)
9. Hansen, J.: Imperative logic and its problems. In: Gabbay, D., Horty, J., Parent, X., van der Meyden, R., van der Torre, L. (eds.) Handbook of Deontic Logic and Normative Systems. College Publications (2013)
10. Humphreys, L., Boella, G., van der Torre, L., Robaldo, L., Di Caro, L., Ghanavati, S., Muthuri, R.: Populating legal ontologies using semantic role labeling. Artificial Intelligence and Law **29**(2) (2021)
11. Jörgensen, J.: Imperatives and logic. Erkenntnis **7**(1) (1937)
12. Kelsen, H.: Conflicts of Norms. In: General Theory of Norms. Oxford University Press (1991)
13. Lam, H., Governatori, G.: The making of spindle. In: Governatori, G., Hall, J., Paschke, A. (eds.) Proc. of Rule Interchange and Applications (RuleML 2009). Lecture Notes in Computer Science, vol. 5858. Springer (2009)
14. Liga, D., Robaldo, L.: Fine-tuning GPT-3 for legal rule classification. Computer Law & Security Review **51**, 105864 (2023)
15. Makinson, D., van der Torre, L.W.N.: Input/output logics. Journal of Philosophical Logic **29**(4), 383–408 (2000)
16. Parent, X., van der Torre, L.: "sing and dance!" - input/output logics without weakening. In: Proc. of Deontic Logic and Normative Systems - 12th International Conference, DEON 2014 (2014)
17. Robaldo, L., Bartolini, C., Palmirani, M., Rossi, A., Martoni, M., Lenzini, G.: Formalizing gdpr provisions in reified i/o logic: the dapreco knowledge base. The Journal of Logic, Language, and Information **29** (2020)
18. Robaldo, L., Batsakis, S., Calegari, R., Calimeri, F., Fujita, M., Governatori, G., Morelli, M., Pacenza, F., Pisano, G., Satoh, K., Tachmazidis, I., Zangari, J.: Compliance checking on first-order knowledge with conflicting and compensatory norms: a comparison among currently available technologies. Artificial Intelligence and Law **32**(2) (2024)
19. Robaldo, L., Pacenza, F., Zangari, J., Calegari, R., Calimeri, F., Siragusa, G.: Efficient compliance checking of RDF data. Journal of Logic and Computation **33**(8) (2023)
20. Robaldo, L., Pozzato, G.: Handling irresolvable conflicts in the Semantic Web: an RDF-based conflict-tolerant version of the Deontic Traditional Scheme. https://arxiv.org/abs/2411.19918 (2024)
21. Sun, X., Robaldo, L.: On the complexity of input/output logic. The Journal of Applied Logic **25** (2017)
22. Vranes, E.: The Definition of 'Norm Conflict' in International Law and Legal Theory. European Journal of International Law **17**(2) (2006)
23. von Wright, G.: Deontic logic. Mind **60** (1951)