Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020), pages 5688–5697
Marseille, 11–16 May 2020

The DAPRECO knowledge base: representing the GDPR in LegalRuleML

Livio Robaldo, Cesare Bartolini, Gabriele Lenzini
University of Luxembourg
2, avenue de l'Université, 4365 Esch-sur-Alzette, Luxembourg
{livio.robaldo, cesare.bartolini, gabriele.lenzini}@uni.lu

Abstract

The DAPRECO knowledge base (D-KB) is a repository of rules written in LegalRuleML, an XML formalism designed to represent the logical content of legal documents. The rules represent the provisions of the General Data Protection Regulation (GDPR). The D-KB builds upon the Privacy Ontology (PrOnto) (Palmirani et al., 2018a), which provides a model for the legal concepts involved in the GDPR, by adding a further layer of constraints in the form of if-then rules, referring either to standard first order logic implications or to deontic statements. If-then rules are formalized in reified I/O logic (Robaldo and Sun, 2017) and then codified in (LegalRuleML, 2019). To date, the D-KB is the biggest knowledge base in LegalRuleML freely available online at (Robaldo et al., 2019).

Keywords: Reification, LegalRuleML, GDPR

1. Introduction

In the years 2010-2016, research in Computational Law had primarily focused on the application of NLP methods to legal texts, for designing legal document management systems to assist legal professionals in retrieving the information they are interested in. An example is the Eunomos legal document management system (Boella et al., 2016).

Eunomos and similar systems classify, index, and discover inter-links between legal documents, possibly by employing NLP techniques (see (Robaldo et al., 2011), (Boella et al., 2012), and (Adebayo et al., 2016). This is often done by transforming the source legal documents into XML standards, such as Akoma Ntoso (Palmirani and Vitali, 2011), and tagging the relevant information. Subsequent phases are devoted to archiving and querying the XML files.

Documents are tagged with respect to legal ontologies, e.g., (Ajani et al., 2017). These encode formal naming and definitions of the concepts involved in the modeled domain, which enables reuse and cross-document navigation and search. The ontological concepts may also be linked to other concepts from external public ontologies from the Web of Things, including Linked Open Data (LOD), thus enhancing the interoperability, the standardization, and the reasoning capabilities of the resources.

Although the joint use of Akoma Ntoso and legal ontologies indeed helps navigate legislation, its overall usefulness is limited due to the focus on terminological issues and information retrieval, all the while disregarding the specific semantic aspects of law. Determining what is obligatory, permitted, or forbidden, which obligations have been fulfilled or violated, and which ones are still in force, allow inferences that can be expended in decision making.

For this reason, recent research in Computational Law led to the identification of a new level devoted to *logic rules*, as exemplified in Figure 1. Systems such as Eunomos incorporate the first two levels in Figure 1, while the third one (logic rules) is still at the stage of basic research.

A new standardization initiative called (LegalRuleML, 2019) has been recently proposed to explicitly deal with this new level. LegalRuleML separately represents and stores the logical content of the provisions, while associating them with both the paragraphs of the Akoma Ntoso documents and the concepts in the legal ontology.

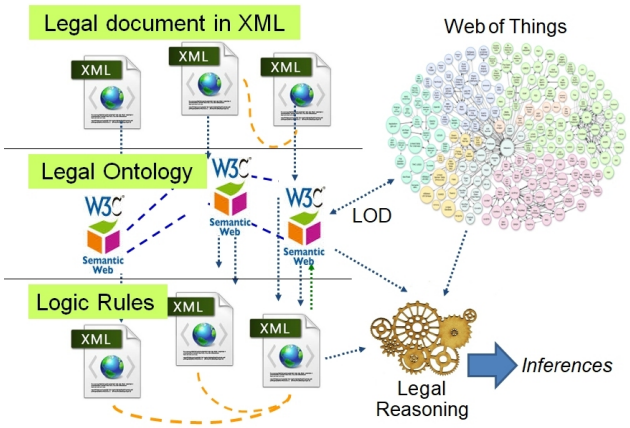

Figure 1: Three levels in Computational Law

LegalRuleML provides a set of XML tags to encode formulæ in some logic. In this paper, we will consider reified I/O logic (Robaldo and Sun, 2017), a novel formalism to represent legal provisions in natural language.

This paper presents an implementation, formalized in reified I/O logic and encoded in LegalRuleML, of the third level in Figure 1, with respect to the GDPR.

This is the DAPRECO knowledge base (D-KB), the main tangible output of the DAPRECO (DAta Protection REgulation COmpliance) research project (Bartolini et al., 2016). The D-KB is built upon recent research results obtained in the context of the (MIREL, 2019) project, which cover the first two levels in Figure 1, with respect to the GDPR:

- The GDPR has been tagged in Akoma Ntoso. The indexes of the structural elements (paragraphs, points, etc.) of the Akoma Ntoso file are used within the D-KB in order to associate these elements with the reified I/O logic formulæ representing their meaning.

- An ontology called PrOnto (Privacy Ontology) has been developed in `OWL2-DL` (Palmirani et al., 2018a), (Palmirani et al., 2018b). PrOnto concepts are

associated with the predicates used in the formulæ in the D-KB, via LegalRuleML.

PrOnto has been built via standard minimization principles from the ontology engineering literature (Brank et al., 2005; Bandeira et al., 2016). Thus, PrOnto does not truly fit for legal reasoning: PrOnto only defines the main concepts involved in the GDPR and their inter-relations, but it lacks another component necessary for legal reasoning: *defeasibility* (cf. (Casini et al., 2015)).

Reified I/O logic has been precisely designed to represent deontic statements in natural language, possibly defeasible, by means of a simple formal machinery that facilitates the creation of large knowledge bases of formulæ.

The next sections, after a quick introduction of reified I/O logic, will focus on some of the main Natural Language Semantics issues encountered during the development of the D-KB, and how reified I/O logic is able to cope with them. On the other hand, since XML is rather verbose, space constraints forbid us to report the corresponding LegalRuleML representations in this paper; however, these are publicly available online at (Robaldo et al., 2019).

Overall, the D-KB includes 966 formulæ formalized in reified I/O logic and encoded in LegalRuleML. It thus represents a benchmark for the XML legal standard: no other so large repositories of LegalRuleML representations are publicly available online. In future works, we plan to conduct further research on the top of the D-KB and to develop applications in the data protection domain that use it.

2. Reified I/O logic

Reified I/O logic (Robaldo and Sun, 2017) is a recent novel formalism for representing norms in textual form. It combines I/O logic (Makinson and van der Torre, 2000) with the reification-based logic in (Hobbs and Gordon, 2017).

2.1. I/O logic

I/O logic is a formalism for deontic reasoning. Unlike logics based on possible-world semantics, I/O logic adopts norm-based semantics in the sense of (Hansen, 2014).

I/O systems are families of if-then rules in the form (x, y), such that when x is given in input, y is returned in output. Further axioms may be added to constrain the behavior of the if-then rules. For instance, (Makinson and van der Torre, 2001) define additional meta-structures to handle contrary-to-duty reasoning. Moreover, as (Sun and Robaldo, 2017) shows, I/O logic features a computational complexity lower than other deontic logics, in particular those based on possible-world semantics. These issues are beyond the scope of this paper, so that we address the interested reader to the mentioned references.

(Boella and van der Torre, 2004) was the first attempt to use I/O logic for legal reasoning. The approach is based on the well-known distinction between regulative norms and constitutive norms (Searle, 1995). The former are obligations, permissions, and prohibitions, i.e., the deontic statements. The latter are definitions that model the meaning of the concepts used in the former. Drawing from (Boella and van der Torre, 2004), (Sun and van der Torre, 2014) propose to use *two* sequential I/O systems, as shown in Figure 2:

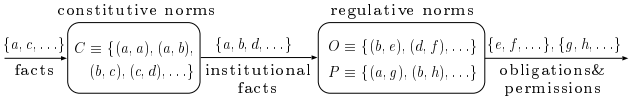

Figure 2: I/O logic system for the legal domain.

The first set C is the set of constitutive norms. Every pair $(x, y) \in C$ is a standard first-order implication '$x \rightarrow y$'. C takes as input the facts of the domain and returns the *institutional* facts. In (Searle, 1995)'s terminology, C defines when something *counts as* something else in the domain.

The output of C is given in input to other two sets of pairs, O and P: the set of obligations and the set of permissions. A pair $(x, y) \in O$ reads as "given x, y is obligatory", while a pair $(x, y) \in P$ reads as "given x, y is permitted".

We can then compute, under different axioms on C, O, and P, what is obligatory, permitted, or forbidden, which obligations have been fulfilled or violated, and which ones are still in force; details in (Makinson and van der Torre, 2000).

2.2. (Hobbs and Gordon, 2017)

(Hobbs and Gordon, 2017)[1] proposes a wide-coverage logic for natural language able to handle a large set of linguistic phenomena into a formally simple formalism. It is grounded on the notion of *reification* (Davidson, 1967).

In (Hobbs and Gordon, 2017), every FOL predication, e.g., '($blond$ `John`)' asserting that John is blond, may be associated with another FOL predication '($blond$' e_b `John`)', where e_b is a new FOL term called "eventuality". e_b is the reification of John's "blond-ness", i.e., it represents *the fact that* John is blond. Other predications may be then applied to e_b, and recursively reified into new eventualities. For instance, we may represent "John wishes to be blond" as:

(1) ($wish$' e_w `John` e_b) $\wedge$ ($blond$' e_b `John`)

In order to distinguish that the fact that John wishes to be blond (variable e_w) holds in the context at time t, while nothing can be inferred about the fact that John is blond (variable e_b), we assert '($RexistAtTime$ e_w t)'. Only eventualities for which $RexistAtTime$ is asserted on t really exist at time t. The final representation is then:

(2) ($RexistAtTime$ e_w t) $\wedge$ ($wish$' e_w `John` e_b) $\wedge$ ($blond$' e_b `John`)

(Hobbs and Gordon, 2017)'s logic is characterized by a massive use of reification, which may be applied to every relation on FOL terms, including boolean operators. For instance, '(not' e_1 e_2)' is used to assert that e_1 is the eventuality of e_2's not existing, while '(or' e e_1 e_2)' states that e is the fact that at least one of e_1 and e_2 really exists.

In order to get the intended meaning, not' and or' need to be defined in terms of additional axioms/definitions: (3) and (4) respectively. (3) states that if two eventualities e and e_1 are related to each other in terms of a not' relation, whenever e really exists, e_1 does not. (4) states that if two eventualities e_1 and e_2 are related with a third eventuality e in terms of an or' relation, whenever e really exists, at least one of e_1 and e_2 really exists too.

[1] See also `http://www.isi.edu/~hobbs/csk.html`.

(3) $\forall_t \forall_e \forall_{e_1} [\ ((RexistAtTime\ e\ t) \wedge (not'\ e\ e_1)) \rightarrow \neg(RexistAtTime\ e_1\ t)\ ]$

(4) $\forall_t \forall_e \forall_{e_1} \forall_{e_2} [\ ((RexistAtTime\ e\ t) \wedge (or'\ e\ e_1\ e_2)) \rightarrow ((RexistAtTime\ e_1\ t) \vee (RexistAtTime\ e_2\ t))\ ]$

Note that not' and or' are not boolean operators. Rather, they are FOL predicates relating two and three eventualities respectively. Then, axioms (3) and (4) define their meaning in terms of the boolean operators '$\neg$' and '$\vee$'.
In this paper, we distinguish between formulae belonging to the assertive contextual statements (ABox), such as (1) and (2), from formulae belonging to the terminological declarative statements (TBox), such as (3) and (4). The former are flat conjunctions of atomic predications, while the latter may be any formula in standard FOL.
With reification, (Hobbs and Gordon, 2017) avoids any nesting of subformulae within complex operators. This is the main insight of the approach: modeling complex linguistic phenomena in terms of *flat* atomic FOL (reified) predicates in the ABox. The definitions of these predicates is separately modeled in terms of axioms in the TBox.
In light of this, combining (Hobbs and Gordon, 2017) and I/O logic appears to be a promising choice. In I/O logic we assert *flat* set of if-then rules, constrained by separate axioms, aiming at modeling the same meaning that standard deontic logic aims at modeling via complex deontic operators that embed (nested) subformulae.

2.3. Combining reification and I/O logic

In reified I/O logic, combines (Hobbs and Gordon, 2017) and I/O logic by taking the elements x and y of every I/O pair (x, y) in Figure 2 to be (Hobbs and Gordon, 2017)'s formulae representing excerpts of NL norms.
In other words, the sets C, O, and P (constitutive rules, obligations, and permissions) defines the ABox of the I/O normative system in Figure 2. Their elements are always pairs of conjunctions of atomic predications. On the other hand, the TBox includes the semantic relations codified in PrOnto, e.g., the is-a relations between the ontological classes associated with the predicates, as well as definitions of other needed predicates, such as (3) and (4) above for handling negation and disjunction. Note that PrOnto does not add complexity to the overall system, in that it is written in `OWL2-DL`, which is a decidable fragment of FOL.
A first example is the obligation in (5), formalized as in (6):

(5) Those who are not wearing a tie or those who are blond ought to leave the room.

(6) $\forall_x \forall_t ($
$\exists_{e_o} \exists_{e_n} \exists_{e_b} \exists_{e_w} \exists_{t_1} [(RexistAtTime\ e_o\ t) \wedge$
$(or'\ e_o\ e_n\ e_b) \wedge (blond'\ e_b\ x) \wedge$
$(not'\ e_n\ e_w) \wedge (wearing'\ e_w\ x\ t_1) \wedge (tie\ t_1)],$
$\exists_{e_l} [(RexistAtTime\ e_l\ t) \wedge (leave'\ e_l\ x\ \mathtt{R})]\ ) \in O$

In (6), universal quantifiers external to the pair are introduced in order to "carry" single individuals from the input to the output. In other words, they act as a "bridge" from the left side to the right side. Only variables which occur in both sides are bound by these quantifiers.

(6) reads as follows: for every x and for every time t, if at t either x does not wear a tie t_1 or x is blond, then it is obligatory at time t the real existence of a "leaving" action e_l from the room `R`, performed by x.
A more complex example, taken from the D-KB, is the representation of the provision in Art. 12(7) of the GDPR:

(7) `Art.12(7)`: The information to be provided to data subjects pursuant to Articles 13 and 14 may be provided in combination with standardised icons in order to give in an easily visible, intelligible and clearly legible manner a meaningful overview of the intended processing. Where the icons are presented electronically they shall be machine-readable.

(7) contains both a permission and an obligation, formalized, in the D-KB, as formulæ (8) and (9) respectively.

(8) $\forall_{t_1} \forall_y \forall_{e_n} ($
$\exists_{a_1} \exists_{e_p} \exists_{e_{dp}} \exists_w \exists_z \exists_x \exists_i [\ (RexistAtTime\ a_1\ t_1) \wedge$
$(and'\ a_1\ e_p\ e_n\ e_{dp}) \wedge (DataSubject\ w) \wedge$
$(PersonalData\ z\ w) \wedge (Controller\ y\ z) \wedge$
$(Processor\ x) \wedge (nominates'\ e_{dp}\ y\ x) \wedge$
$(PersonalDataProcessing'\ e_p\ x\ z) \wedge$
$(Communicate'\ e_n\ y\ w\ i)],$
$\exists_{eat} \exists_{ic} [(RexistAtTime\ e_{at}\ t_1) \wedge$
$(AttachTo'\ e_{at}\ y\ ic\ e_n) \wedge (Icon\ ic)]\ ) \in P$

(9) $\forall_{t_1} \forall_{ic} (\exists_{a_1} \exists_{e_p} \exists_{e_n} \exists_{e_{at}} \exists_{e_l} \exists_{e_{dp}} \exists_w \exists_z \exists_y \exists_x \exists_i [$
$(RexistAtTime\ a_1\ t_1) \wedge$
$(and'\ a_1\ e_p\ e_n\ e_{at}\ e_l\ e_{dp}) \wedge (DataSubject\ w) \wedge$
$(PersonalData\ z\ w) \wedge (Controller\ y\ z) \wedge$
$(Processor\ x) \wedge (nominates'\ e_{dp}\ y\ x) \wedge$
$(PersonalDataProcessing'\ e_p\ x\ z) \wedge$
$(Communicate'\ e_n\ y\ w\ i) \wedge (Icon\ ic) \wedge$
$(electrForm'\ e_l\ ic) \wedge (AttachTo'\ e_{at}\ y\ ic\ e_n)],$
$\exists_{e_{mr}} [(RexistAtTime\ e_{mr}\ t_1) \wedge$
$(machineReadableness'\ e_{mr}\ ic)]\ ) \in O$

Formula (8) contains an and' predicate[2]. This is a relation between multiple eventualities: its first argument (a_1, in (8)) really exists if and only if all other arguments (e_p, e_n, and e_{dp}, in (8)) really exist. All other predicates in (8) parallel NL words and do not need particular explanations. (8) reads as follows: if there is a processor y and a notification event e_n of some information i, performed by y with respect to a data subject w (to whom some personal data z are related, processed by a processor x and controlled by y), then y is permitted to attach an icon ic to e_n.
On the other hand, (9) states that in case the icon is in electronic form, it ought to be machine-readable.
A significant difference between (9) and (8) is that the former also requires the real existence of e_{at} and of e_l. e_{at} refers to the action of attaching an icon to the notification, performed by the controller y, while e_l refers to the fact

[2]See `https://www.isi.edu/~hobbs/bgt-logic.text` for details.

that the icon ic is in electronic form. In case e_{at} and e_l really exist as well, then the "machine-readable-ness" of ic is obligatorily required (eventuality e_{mr}).

3. Building the D-KB

The current version of the D-KB includes 966 formulæ in reified I/O logic: 271 obligations, 76 permissions, and 619 constitutive rules. The number of constitutive rules is much higher than the one of obligations and permissions as the D-KB includes constitutive rules needed to trigger the special inferences explained below in subsection 3.1.

The formulæ refer to the paragraphs in the GDPR. It is possible to associate more than one formula with a paragraph. For instance, Art. 12(7), shown above in (7), is associated with both an obligation and a permission ((8) and (9)).

On the other hand, let us consider an example of if-then rule belonging to the set C. The rule represents the meaning of Article 6, paragraph 1, point 1, of the GDPR.

According to Art. 5(1)(a) of the GDPR, it is obligatory for a processing of personal data to be *lawful*. This is mirrored in the PrOnto ontology via a boolean attribute called LAWFULNESS, which is a data property of a PrOnto class PERSONALDATAPROCESSING. The attribute is true when the processing is lawful, and false otherwise. However, PrOnto does not specify the conditions under which the LAWFULNESS attribute is either true or false. This is done by means of a set of formulæ in the D-KB, specifically in the set C.

For instance, Art. 6(1)(a), of the GDPR specifies that the processing is lawful if "*the data subject has given consent to the processing of his or her personal data for one or more specific purposes;*". This is formalized[3] in formula (10).

(10) $\forall_{e_p}(\exists_{a_1}\exists_{t_1}\exists_{e_{hc}}\exists_{e_{au}}\exists_{e_{dp}}\exists_{w}\exists_{z}\exists_{y}\exists_{x}\exists_{c}[$
$(RexistAtTime\ a_1\ t_1) \wedge (and'\ a_1\ e_p\ e_{hc}\ e_{au}\ e_{dp}) \wedge$
$(DataSubject\ w) \wedge (PersonalData\ z\ w) \wedge$
$(Controller\ y\ z) \wedge (Purpose\ e_{pu}) \wedge$
$(nominates'\ e_{dp}\ y\ x) \wedge (Processor\ x) \wedge$
$(PersonalDataProcessing'\ e_p\ x\ z) \wedge$
$(isBasedOn\ e_p\ e_{pu}) \wedge (GiveConsent'\ e_{hc}\ w\ c) \wedge$
$(Consent\ c) \wedge (AuthorizedBy'\ e_{au}\ e_{pu}\ c)$],
$(lawfulness\ e_p)\ ) \in C$

Constitutive rules behave as standard FOL implications; in case of (10), if the left side holds in the context, "$(lawfulness\ e_p)$" (and the corresponding LAWFULNESS attribute of PrOnto) is true. As said above, the connection between the predicates in the formulæ and the corresponding PrOnto concepts is realized by LegalRuleML tags.

We developed a JavaScript tool to guide and monitor the building of the D-KB formulæ, see Figure 3. The tool allows to load an Akoma Ntoso file, select an excerpt of text, and associate it with one or more reified I/O logic formulæ. Since reified I/O logic formulæ are if-then rules of conjunctions of atomic predications, the human annotator simply specifies, one by one, the predicates in the conjunctions,

[3]Indeed, (10) lacks a predicate referring to an exception to Art. 6(1)(a). The handling of exceptions will be illustrated below in subsection 3.2., so that we avoid that predicate in (10).

Figure 3: The tool to edit reified I/O logic formulæ

while filling them with their arguments. Special placeholders allow to specify whether the rule is an obligation, a permission, or a constitutive rule. Once the formulæ are complete, the tool allows to save the result in LegalRuleML, while automatically creating the associations with the Akoma Ntoso indexes and the PrOnto concepts. In future works, we plan to extend and reuse the tool for translating in reified I/O logic other legislative documents.

Since it is not possible to illustrate the details of each formula in the D-KB, below we only explain how reification allows to easily deal with three well-known thorny issues for the representations of NL norms: nested obligations and permissions, defeasibility, and legal interpretations.

3.1. Nested obligations and nested permissions

Reification allows to avoid nestings of subformulæ within complex operators. The latter are represented in terms of FOL predicates that take as argument the main eventuality referring to the state of affairs described by the subformula. Similar nestings are found also in deontic assertions, although they are rare, and they have been addressed, for instance, in (Governatori, 2015). (11) shows an example of"nested obligations"'; similar patterns can be of course found for permissions.

(11) If a manager is obliged to perform an action a, his secretary is obliged to write it down in his agenda.

The intended meaning of (11) is that the secretary is obliged to write down in the manager's agenda the fact that the manager is obliged to perform the action a. In standard I/O logic it is not possible to represent the meaning of nested obligations such as (11), as the formalism does not allow to specify semantic links between the if-then rules. Conversely, reification makes that straightforwardly possible.

Suppose that every manager attending a party "p" is obliged to be elegant, represented as in (12):

(12) $\forall_m\forall_t(\exists_{e_a}[(RexistAtTime\ e_a\ t) \wedge$
$(attend'\ e_a\ m\ \mathrm{p}) \wedge (manager\ m)],$
$\exists_{e_e}[(RexistAtTime\ e_e\ t) \wedge (elegant'\ e_e\ m)]\ ) \in O$

We can then represent the status of being obliged in terms of a new predication '$(Obliged'\ e_o\ t\ m\ e)$', meaning that m is obliged to the real existence of e at time t. This is enforced by adding a special constitutive rule, associated with (12), asserting that if a manager m is attending $\mathtt{p}$, then s/he is in the status of being obliged to be elegant:

(13) $\forall_m \forall_t (\ \exists_{e_a} [(RexistAtTime\ e_a\ t) \wedge (attend'\ e_a\ m\ \mathtt{p}) \wedge (manager\ m)],$
$\exists_{e_o} [(Obliged'\ e_o\ t\ m\ e_e) \wedge (elegant'\ e_e\ m)]\ ) \in C$

The obligation of the secretary is then represented as:

(14) $\forall_m \forall_{e_o} \forall_t (\ \exists_e [(Obliged'\ e_o\ t\ m\ e) \wedge (manager\ m)],$
$\exists_{e_w} [(RexistAtTime\ e_w\ t) \wedge (write'\ e_w\ \mathtt{secr}(m)\ e_o)]\ ) \in O$

In (14), '$\mathtt{secr}(m)$' is a FOL function that, taken a manager m, returns his secretary, who is the agent of the writing action e_w. On the other hand, the object/patient of e_w is the fact that m is obliged to the real existence of e at time t.
Like $RexistAtTime$, $Obliged$ is a possible modality that eventualities may hold. $RexistAtTime$ specifies which eventualities hold the status of real existence at t, while $Obliged$ specifies which ones hold the status of obligatoriness at t. A dual predicate $Permitted$ is instead used for eventualities that hold the status of permissiveness at t.
In light of this, every obligation and permission is associated with an additional constitutive rule specifying the status of obligatoriness or permissiveness of their bearers. This also applies recursively; for instance, (15), associated with (14), states that secretary is in the status of being obliged to write down the obligations of the manager:

(15) $\forall_m \forall_{e_{o1}} \forall_t (\exists_e [(Obliged'\ e_{o1}\ t\ m\ e) \wedge (manager\ m)],$
$\exists_{e_{o2}} \exists_{e_w} [(Obliged'\ e_{o2}\ t\ \mathtt{secr}(m)\ e_w) \wedge (write'\ e_w\ \mathtt{secr}(m)\ e_{o1})]\ ) \in C$

As said above, although nested obligations are rare, they could indeed occur in existing legislation. The D-KB formalizes some as well, e.g., Art. 17(2) of the GDPR:

(16) `Art.17(2)`: Where the controller has made the personal data public and is obliged pursuant to par. 1 to erase the personal data, the controller, taking account of available technology and the cost of implementation, shall take reasonable steps, including technical measures, to inform controllers which are processing the personal data that the data subject has requested the erasure by such controllers of any links to, or copy or replication of, those personal data.

(16) is represented via the reified I/O logic formula in (17):

(17) $\forall_{y_1} \forall_{y_2} \forall_w \forall_z \forall_{t_1} (\exists_{e_{ob}} \exists_{e_{ra1}} [(RexistAtTime\ e_{ob}\ t_1) \wedge (DataSubject\ w) \wedge (PersonalData\ z\ w) \wedge (Controller\ y_1\ z) \wedge (Obliged'\ e_{ob}\ y_1\ e_{ra1}\ t_1) \wedge (Delete'\ e_{ra1}\ w\ z) \wedge (public\ z) \wedge (Controller\ y_2\ \mathtt{copyOf}(z))\ ],$
$\exists_{e_n} \exists_{e_{ra2}} \exists_{t_2} [(RexistAtTime\ e_n\ t_2) \wedge (numeric\text{-}greater\text{-}than\text{-}or\text{-}equal\ t_2\ t_1) \wedge (Communicate'\ e_n\ y_1\ y_2\ e_{ra2}) \wedge (Delete'\ e_{ra2}\ w\ \mathtt{copyOf}(z)) \wedge (reasonable\ e_n)]\ ) \in O$

(17) reads as follows: if a controller y_1 controls public personal data z of w, a controller y_2 controls a copy of z, and y_1 is obliged to make e_{ra1} really existing (where e_{ra1} is the fact that w has requested to erase z), then, in $t_2 \geq t_1$, y_1 is obliged to communicate e_{ra2} to y_2. In other words, if y_1 is obliged to delete, he is (also) obliged to communicate. Furthermore, such a communication must be "reasonable"[4].

3.2. Exceptions

It is often the case that *general* regulative norms are overridden by more *specific* rules in restricted contexts. Those more specific rules are then *exceptions* to the general rule. Furthermore, specific rules may be in turn defined in the restricted contexts (exceptions of exceptions). An example from Art. 8(1) of the GDPR is shown in (18):

(18) `Art. 8(1)`: Where Art. 6(1)(a) applies, in relation to the offer of information society services directly to a child, the processing of the personal data of a child shall be lawful where the child is at least 16 years old. Where the child is below the age of 16 years, such processing shall be lawful only if and to the extent that consent is given or authorised by the holder of parental responsibility over the child. Member States may provide by law for a lower age for those purposes provided that such lower age is not below 13 years.

Art. 8(1) refers to Art. 5(1)(a) and Art. 6(1)(a):

(19) `Art. 5(1)(a)`: Personal data shall be: (a) processed lawfully, fairly and in a transparent manner in relation to the data subject ... [*CUT*];

`Art. 6(1)(a)`: Processing shall be lawful only if and to the extent that at least one of the following applies: (a) the data subject has given consent to the processing of his or her personal data for one or more specific purposes;

The meaning of (18) may be summarized as follows:

(20)
- Processing of personal data ought to be lawful.
- If the data subject has given consent to the processing, then it is lawful.
- Exception to (b): if the data subject is less than 16 years old, his consent does not entail lawfulness of processing.
- If the data subject is less than 16 years old and if the holder of his parental responsibility has given consent to the processing, then it is lawful.
- Exception to (c)-(d): Member States may lower the minimal age for consent down to 13 years.

[4]The truth value of '$reasonable$' depends on the legal interpretation ascribed to the associated adjective; see subsection 3.3..

The formulæ representing (20.a–b) are shown in (21) and (22). Note that (21) is an obligation, while (22) is a constitutive rule, used to define the lawfulness of processing.

(21) $\forall_{e_p}\forall_t(\exists_{a_1}\exists_{e_{dp}}\exists_x\exists_y\exists_z\exists_w[\ (RexistAtTime\ a_1\ t) \wedge (and\ a_1\ e_{dp}\ e_p) \wedge (DataSubject\ w) \wedge (PersonalData\ z\ w) \wedge (Controller\ y\ z) \wedge (nominates'\ e_{dp}\ y\ x) \wedge (Processor\ x) \wedge (PersonalDataProcessing'\ e_p\ x\ z)\ ], \exists_{e_l}[(RexistAtTime\ e_l\ t) \wedge (lawfulness'\ e_l\ e_p)]) \in O$

(22) $\forall_{e_p}\forall_t(\exists_{epu}\exists_{e_{dp}}\exists_{e_{hc}}\exists_{e_{au}}\exists_{a_1}\exists_x\exists_y\exists_z\exists_w\exists_c[(RexistAtTime\ a_1\ t) \wedge (and\ a_1\ e_{dp}\ e_p\ e_{hc}\ e_{au}) \wedge (DataSubject\ w) \wedge (PersonalData\ z\ w) \wedge (Controller\ y\ z) \wedge (nominates'\ e_{dp}\ y\ x) \wedge (Processor\ x) \wedge (Purpose\ e_{pu}) \wedge (PersonalDataProcessing'\ e_p\ x\ z) (isBasedOn\ e_p\ e_{pu}) \wedge (GiveConsent'\ e_{hc}\ w\ c) \wedge (Consent\ c) \wedge (AuthorizedBy'\ e_{au}\ e_{pu}\ c)], \exists_{e_l}[(RexistAtTime\ e_l\ t) \wedge (lawfulness'\ e_l\ e_p)]) \in C$

(20.c) is represented via the constructs proposed in (Hobbs and Gordon, 2017)[5] to implement *defeasibility*, which are drawn from Circumscriptive Logic (McCarthy, 1980).
We represent general inferences while adding special predicates to "block" them in case of exception. Thus, general inferences are allowed only if more specific exceptions do *not* occur. Specifically, we use negation-as-failure, written as $naf(X)$, which is supported in LegalRuleML. The predication $naf(X)$ is true when it cannot be derived that X is true, i.e., when it is either false or *unknown*.
Formula (22) is then rewritten as (23). The only difference between the two formulæ is the predication '$naf(exceptionCha2Art8Par1\ e_p)$', which is true if the exception about the processing e_p does not occur.

(23) $\forall_{e_p}\forall_t(\exists_{epu}\exists_{e_{dp}}\exists_{e_{hc}}\exists_{e_{au}}\exists_{a_1}\exists_x\exists_y\exists_z\exists_w\exists_c[(RexistAtTime\ a_1\ t) \wedge (and\ a_1\ e_{dp}\ e_p\ e_{hc}\ e_{au}) \wedge (DataSubject\ w) \wedge (PersonalData\ z\ w) \wedge (Controller\ y\ z) \wedge (nominates'\ e_{dp}\ y\ x) \wedge (Processor\ x) \wedge (Purpose\ e_{pu}) \wedge (PersonalDataProcessing'\ e_p\ x\ z) (isBasedOn\ e_p\ e_{pu}) \wedge (GiveConsent'\ e_{hc}\ w\ c) \wedge (Consent\ c) \wedge (AuthorizedBy'\ e_{au}\ e_{pu}\ c) \wedge naf(exceptionCha2Art8Par1\ e_p)], \exists_{e_l}[(RexistAtTime\ e_l\ t) \wedge (lawfulness'\ e_l\ e_p)]) \in C$

(20.c) can then be modeled via (24), which entails '$(exceptionCha2Art8Par1\ e_p)$', in case the data subject is less than 16 years old, and "blocks" the inference in (23).

(24) $\forall_{e_p}(\ \exists_{a_1}\exists_t\exists_{e_{dp}}\exists_x\exists_y\exists_z\exists_w[(RexistAtTime\ a_1\ t) \wedge (and\ a_1\ e_p\ e_{dp}) \wedge (DataSubject\ w) \wedge (PersonalData\ z\ w) \wedge (Controller\ y\ z) \wedge (nominates'\ e_{dp}\ y\ x) \wedge (Processor\ x) \wedge (PersonalDataProcessing'\ e_p\ x\ z) (numeric\text{-}less\text{-}than\ \texttt{ageOf}(w)\ 16)], (exceptionCha2Art8Par1\ e_p)\ ) \in C$

New obligations may be then consistently asserted for data subjects who are less than 16 years old, e.g., the formula representing (20.d), which is omitted for space constraints.
Finally, it is necessary to (recursively) define an exception in (24) as well, since Art. 8(1) allows Member States to lower the minimal age for consent (although not below 13 years). To avoid over-assertion of rules, we can substitute the constant '16' in (24) with a FOL function '`minAgeConsent`' that associate the processing e_p with the minimal age for giving consent to e_p in the context where the formula is evaluated. We then replace '$(numeric\text{-}less\text{-}than\ \texttt{ageOf}(w)\ 16)$' in (24) with '$(numeric\text{-}less\text{-}than\ \texttt{ageOf}(w)\ \texttt{minAgeConsent}(e_p))$'. The following constitutive rule defines that the value of minimal age for consent is 16, provided that there are no exceptions:

(25) $\forall_{e_p}(naf(exceptionMinAgeConsent\ e_p), (numeric\text{-}equal\ \texttt{minAgeConsent}(e_p)\ 16)) \in C$

Assuming, for instance, that French national legislation lowers the minimal age for consent down to 14 years, the D-KB will be enriched with the following constitutive rule:

(26) $\forall_{e_p}(\ \exists_{a_1}\exists_t\exists_{e_{dp}}\exists_x\exists_y\exists_z\exists_w[(RexistAtTime\ a_1\ t) \wedge (and\ a_1\ e_{dp}\ e_p) \wedge (DataSubject\ w) \wedge (PersonalData\ z\ w) \wedge (Controller\ y\ z) \wedge (nominates'\ e_{dp}\ y\ x) \wedge (Processor\ x) \wedge (PersonalDataProcessing'\ e_p\ x\ z) (equal\ \texttt{MemberState}(y)\ \texttt{France})], (numeric\text{-}equal\ \texttt{minAgeConsent}(e_p)\ 14)\ ) \in C$

3.3. Legal interpretations

Legislators tend to use vague terms, e.g., "reasonable" in (16), to make the norms applicable in a multitude of contexts. It is then up to judges and other appointed authorities to interpret these terms in each context. However, even in similar contexts, it is common that different judges adopt incompatible interpretations (sometimes even concerning identical cases). The D-KB allows these incompatible interpretations to coexist in certain time spans via the LegalRuleML tags specifically designed for this purpose.
We show an example from (Bartolini et al., 2016), which provides some possible legal interpretations of GDPR norms in terms of *correlations* between them and the controls in some ISO security standards. For instance, the D-KB assumes that (27.a-b) are correlated, so that compliance with control A9.1 of the ISO/IEC 27018:2014 security standard in entails compliance with Art. 33(2) of the GDPR:

[5]For a quick explanation, see `https://www.isi.edu/~hobbs/bgt-defeasibility.text`.

(27)
- `GDPR, Art.33(2)`: The processor shall notify the controller without undue delay after becoming aware of a personal data breach.
- `ISO/IEC 27018:2014, A9.1`: The public cloud PII processor should promptly notify the relevant cloud service customer in the event of any unauthorized access to PII.

(27.a-b) are formalized as in (28) and (29) respectively:

(28) $\forall_x\forall_y\forall_{e_b}\forall_{t_1}(\ \exists_{a_1}\exists_{e_{dp}}\exists_{e_p}\exists_{e_a}\exists_w\exists_z[$
$(RexistAtTime\ a_1\ t_1) \wedge (and\ a_1\ e_{dp}\ e_p\ e_a) \wedge$
$(DataSubject\ w) \wedge (PersonalData\ z\ w) \wedge$
$(Controller\ y\ z) \wedge (nominates'\ e_{dp}\ y\ x) \wedge$
$(Processor\ x) \wedge (PersonalDataProcessing'\ e_p\ x\ z) \wedge$
$(AwareOf'\ e_a\ x\ e_b) \wedge (DataBreach\ e_b\ z)]$,
$\exists_{e_n}\exists_{t_2}[(RexistAtTime\ e_n\ t_2) \wedge (nonDelayed\ e_n) \wedge$
$(numeric\text{-}greater\text{-}than\text{-}or\text{-}equal\ t_2\ t_1) \wedge$
$(Communicate'\ e_n\ x\ y\ \texttt{allInfoAbout}(e_b))]) \in O$

(29) $\forall_x\forall_y\forall_{e_a}\forall_{t_1}(\ \exists_{a_1}\exists_{e_{dp}}\exists_{e_p}\exists_w\exists_z\exists_k[$
$(RexistAtTime\ a_1\ t_1) \wedge (and\ a_1\ e_{dp}\ e_p\ e_a) \wedge$
$(PIIPrincipal\ w) \wedge (PIIController\ y\ z) \wedge$
$(PIIProcessor\ x) \wedge (nominates'\ e_{dp}\ y\ x) \wedge$
$(PII\ z\ w) \wedge (PersonalDataProcessing'\ e_p\ x\ z) \wedge$
$(access'\ e_a\ k\ z) \wedge (unauthorized\ e_a)]$,
$\exists_{e_n}\exists_{t_2}[(RexistAtTime\ e_n\ t_2) \wedge (promptly\ e_n) \wedge$
$(numeric\text{-}greater\text{-}than\text{-}or\text{-}equal\ t_2\ t_1) \wedge$
$(Communicate'\ e_n\ x\ y\ \texttt{allInfoAbout}(e_a))]) \in O$

To correlate (29) and (28) we introduce new entailments between them, so that one obligation is satisfied/violated if the other one is. It seems rather unquestionable to assume that "PII" (Personally Identifiable Information), from ISO/IEC 27018:2014, and "personal data", from the GDPR, refer to the same concept. Their definitions are essentially identical. Thus, the D-KB may safely include the constitutive rule: '$\forall_z\forall_w((PII\ z\ w), (PersonalData\ z\ w)) \in C$'. Similarly, we may add to the D-KB the following formulæ, stating that the PIIPrincipal of the online service *is a* data subject, the PII processor *is a* processor, the PII controller *is a* controller, and that "promptly" entails "non-delayed":

- $\forall_w((PIIPrincipal\ w), (DataSubject\ w)) \in C$
- $\forall_x((PIIProcessor\ x), (Processor\ x)) \in C$
- $\forall_y\forall_z((PIIController\ y\ z), (Controller\ y\ z)) \in C$
- $\forall_e((promptly\ e), (nonDelayed\ e)) \in C$

On the other hand, it could be questionable to assume that an unauthorized access *is a* data breach: The constitutive rule in (30) may be subject to different legal interpretations:

(30) $\forall_{e_a}\forall_z\forall_t($
$\exists_k[(RexistAtTime\ e_a\ t) \wedge (access'\ e_a\ k\ z) \wedge$
$(unauthorized\ e_a)]$,
$(DataBreach\ e_a\ z)\ ) \in C$

Similarly to exceptions, we model different legal interpretations by adding a special predicate '$(assumption\ \ e_a)$', which is true if it may be *assumed* that (30) is valid:

(31) $\forall_{e_a}\forall_z\forall_t($
$\exists_k[(RexistAtTime\ e_a\ t) \wedge (access'\ e_a\ k\ z) \wedge$
$(unauthorized\ e_a) \wedge (assumption\ e_a)]$,
$(DataBreach\ e_a\ \ z)\ ) \in C$

The D-KB then contains further constitutive rules to *block* the inference in (30), i.e., to entail '$\neg(assumption\ e_a)$'. It is also possible to introduce constitutive rules that states $(assumption\ e_a)$ as *true*, even if they will be redundant as $(assumption\ e_a)$ is already *assumed* to be true. As argued in (Robaldo and Sun, 2017), the D-KB should be "able to keep track of the different legal interpretations over time". If some legal authorities explicitly state that the default assumptions are true, we should "register" this in terms of parallel explicit formulæ, even if these are redundant. Otherwise, we lose the information that these legal authorities "confirmed" the assumptions. For example, suppose we have the following (fictitious) legal interpretations of (31):

- *Italian Corte di Cassazione*: An unauthorized access *is a* data breach, according to the definitions found in the state of the art of the cybersecurity domains.
- *Spanish Audiencia de Toledo*: An unauthorized access is *not* a data breach, in that a data breach requires not only an unauthorized access, but also a breach of security and a causal connection between them.
- *French Tribunal d'Avignon*: In this case, we assume the Tribunal examined the company "Alpha" performing a security test on an IT system; even if unauthorized accesses indeed took place, they cannot be taken as data breaches in that they were part of the security test.

The three legal interpretations above are modeled via the constitutive rules in (32), (33), and (34) respectively.

(32) $\forall_{e_a}\forall_z\forall_t(\ \exists_k[(RexistAtTime\ e_a\ t) \wedge$
$(access'\ e_a\ k\ z) \wedge (unauthorized\ e_a)]$,
$(assumption\ \ e_a)\ ) \in C$

(33) $\forall_{e_a}\forall_z\forall_t(\ \exists_k[(RexistAtTime\ e_a\ t) \wedge$
$(access'\ e_a\ k\ z) \wedge (unauthorized\ e_a)]$,
$\neg(assumption\ \ e_a)\ ) \in C$

(34) $\forall_{e_a}\forall_z\forall_t(\ \exists_k[(RexistAtTime\ e_a\ t) \wedge$
$(access'\ e_a\ k\ z) \wedge (unauthorized\ e_a) \wedge$
$naf(exceptionSecurityTest\ e_a)]$,
$(assumption\ \ e_a)\ ) \in C$

$\forall_{e_a}(\ \exists_k\exists_z\exists_t[(RexistAtTime\ e_a\ t) \wedge$
$(access'\ e_a\ k\ z) \wedge (unauthorized\ e_a) \wedge$
$(partOf\ e_a\ e_t) \wedge (securityTest\ e_t)]$,
$(exceptionSecurityTest\ \ e_a)\ ) \in C$

(32), (33) and (34) cannot hold together. LegalRuleML provides tags to assert their XOR; we omit their description.

(34) is particularly interesting. It includes *two* constitutive rules, one of which entails an exception to the other one. (34) states that: (1) an unauthorized access is a data breach in the general case; (2) security tests are exceptions to this general case. The difference between (32) and (34) is that, in (32), processors shall comply with GDPR obligations, including the one of notifying the controller without undue delay, also in the case of security tests; on the contrary, in (34), they are exempt from GDPR obligations in such cases.

4. Related works

Reification achieves formal simplicity and modularity, without lowering the expressivity required to represent, for instance, exceptions, legal interpretations, and nested obligations/permissions. This allows for the easy and quick building of large repositories of formulæ, e.g., the D-KB.

Exceptions and legal interpretations have been well-known problems in Computational Law, and several formalizations, e.g., (Sartor, 2005; Governatori et al., 2009; Boella et al., 2010; Rotolo et al., 2015; Walton et al., 2016; Malerba, 2017), have been proposed to deal with them.

For instance, (Rotolo et al., 2015) propose a rule-based framework by adjusting the one proposed in (Governatori and Rotolo, 2008), which is a modal defeasible logic extended with an operator '$\otimes$' to model preferences between multiple legal interpretations. That work adopts three kinds of rules: monotonic implications, non-monotonic (defeasible) implications, and the so-called "defeaters", which are used to "block" certain conclusions; defeaters are similar to our mechanism exemplified for instance in (24). A binary superiority relation '$>$' between rules is then introduced, as well as axioms to constrain their model-theoretic interpretation and the interaction with the '$\otimes$' operator.

The authors then show that their formal machinery can be used to model interpretative arguments in deontic defeasible reasoning in two ways: by interpreting the provisions in their sentential (propositional) meaning as a whole, or by ascribing different legal interpretations to their intra-sentential components, or, in other words, by restricting legal interpretations to the words or the chunks occurring in the textual content describing the provision.

In reified I/O logic, it is not necessary to define two different schemas to distinguish legal interpretations at either the sentential or the intra-sentential level, as reification allows to uniformly move across different levels of abstraction.

On the other hand, nested obligations and nested permissions have been scarcely studied in literature, perhaps because they rarely occur in existing legislation.

A recent paper that addresses them is (Governatori, 2015), where it is shown that Linear Temporal Logic (Pnueli, 1977), used in several contemporary approaches to normative multi-agent systems and business process compliance, is unable to properly deal with nested obligations/permissions. In light of this, that work proposes, as an alternative to Linear Temporal Logic, the special operator originally introduced in (Governatori and Rotolo, 2006), which is not affected by the paradoxes raised by nested obligations/permissions through Linear Temporal Logic inferences.

On the other hand, subsection 3.1. above showed that reification provides a straightforward way to deal with this kind of obligations/permissions in that it has been precisely designed to flatten logical nestings. In fact, it was sufficient to introduce constitutive rules defining the eventualities that refer to "the fact that" someone is obliged/permitted to do something. Then, (nested) obligations/permissions can be *uniformly* asserted on these eventualities and added to the sets O and P, together with the "normal" ones.

5. Conclusions

This paper presented the D-KB, which includes 966 formulæ in reified I/O logic modelling GDPR norms. To date, the D-KB is the biggest knowledge base in I/O logic and LegalRuleML available online, and it aims at becoming a benchmark for the XML standard (Robaldo et al., 2019).

The D-KB has been built in about four months by the first author of this paper, with the aid of a Javascript tool allowing to select the textual span to formalize, build the corresponding formulæ, and save the result in LegalRuleML.

We showed how reified I/O logic allows to express complex legal statements, by presenting examples from the GDPR and other (exemplified) legal statements. Reification allows to avoid nestings and simplify the architecture of the formulæ, thus to build large knowledge bases in a short time.

Indeed, although the LegalRuleML standard has been under design for some years already, no other such large knowledge bases in LegalRuleML is currently available online. There are of course papers showing examples in LegalRuleML, e.g., (Dimyadi et al., 2017), but no enough exhaustive and systemic work has been conducted so far to translate a whole relevant piece of legislation in LegalRuleML. The D-KB is the first achievement in this respect.

In our view, the reason is that the formulæ used in the literature on LegalRuleML, being based on standard embeddings of sub-formulæ within complex operators, are less readable than the reified I/O logic ones, and so harder to edit and debug. As argued in (Robaldo, 2010a), (Robaldo, 2010b), and (Robaldo, 2011), syntactic embeddings are convenient from a formal point of view, in that their model theory may be defined recursively, but they are inadequate for Natural Language Semantics and hard to scale.

Reification allows to straightforwardly represent nested obligations/permissions, by introducing eventualities referring to the fact that someone is obliged/permitted to do something, as well as exceptions and legal interpretations, via special predicates explicitly referring to exceptions and to th assumptions taken in legal interpretations.

Modelling defeasibility with predicates explicitly referring to exceptions and assumptions on certain eventualities appears to be an effective solution for representing legislation. Legislation is normally written by means of *general* and *abstract* provisions, in the sense that legislators know *a priori* only the general contexts where legislation will apply, although sometimes some special situations deserving exceptions to the general rules are known already and encoded in the text of the law. For the most part, legal interpretations will be figured out later; in such cases, it is generally the role of jurisprudence to rule out the correct application of the provisions (case law), but in some cases legislation will be amended in order to account for new exceptions.

Therefore, it seems there is no need to introduce more complex defeasible schema for modeling legislation. Advanced forms of reasoning are instead needed when we do not know *a priori* which rules override other rules, but we have to infer it from the asserted knowledge. This could be needed, for instance, in argumentation systems, where the weight of each argument is assigned at the beginning, then it is inferred which arguments override other ones.

6. Bibliographical References

Adebayo, K. J., Caro, L. D., Robaldo, L., and Boella, G. (2016). Textual inference with tree-structured LSTM. In *28th Benelux Conference on Artificial Intelligence (BNAIC 2016)*, pages 17–31.

Ajani, G., Boella, G., Di Caro, L., Robaldo, L., Humphreys, L., Praduroux, S., Rossi, P., and Violato, A. (2017). The european legal taxonomy syllabus: A multi-lingual, multi-level ontology framework to untangle the web of european legal terminology. *Applied Ontology*, 2 (4).

Bandeira, J., Bittencourt, I. I., Espinheira, P., and Isotani, S. (2016). FOCA: A methodology for ontology evaluation. *CoRR*.

Bartolini, C., Giurgiu, A., Lenzini, G., and Robaldo, L. (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. `https://www.fnr.lu/projects/data-protection-regulation-compliance`.

Boella, G. and van der Torre, L. W. N. (2004). Regulative and constitutive norms in normative multiagent systems. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Ninth International Conference (KR2004)*, pages 255–266.

Boella, G., Governatori, G., Rotolo, A., and van der Torre, L., (2010). *Lex Minus Dixit Quam Voluit, Lex Magis Dixit Quam Voluit: A Formal Study on Legal Compliance and Interpretation*, pages 162–183. Springer Berlin Heidelberg, Berlin, Heidelberg.

Boella, G., di Caro, L., Humphreys, L., Robaldo, L., and van der Torre, L. (2012). NLP Challenges for Eunomos, a tool to build and manage legal knowledge. European Language Resources Association (ELRA).

Boella, G., Di Caro, L., Humphreys, L., Robaldo, L., Rossi, R., and van der Torre, L. (2016). Eunomos, a legal document and knowledge management system for the web to provide relevant, reliable and up-to-date information on the law. *Artificial Intelligence and Law*, 4.

Brank, J., Grobelnik, M., and Mladenić, D. (2005). A survey of ontology evaluation techniques. In *Proceedings of 8th International multi-conference Information Society*.

Casini, G., Meyer, T., Moodley, K., Sattler, U., and Varzinczak, I. (2015). Introducing defeasibility into owl ontologies. In Robert Meersman, et al., editors, *Proc. of International Semantic Web Conference (ISWC)*.

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

Dimyadi, J., Governatori, G., and Amor, R. (2017). Evaluating legaldocml and legalruleml as a standard for sharing normative information in the aec/fm domain. In *Proc. of Joint Conference on Computing in Construction (JC3), Heraklion, Greece, Volume: 1*.

Governatori, G. and Rotolo, A. (2006). Logic of violation: a gentzen system for reasoning with contrary-to-duty obligations. *Australasian Journal of Logic*, (426).

Governatori, G. and Rotolo, A. (2008). Bio logical agents: Norms, beliefs, intentions in defeasible logic. *Autonomous Agents and Multi-Agent Systems*, 17(1).

Governatori, G., Padmanabhan, V., Rotolo, A., and Sattar, A. (2009). A defeasible logic for modelling policy-based intentions and motivational attitudes. *Logic Journal of the IGPL*, 17(3).

Governatori, G. (2015). Thou shalt is not you will. In *Proceedings of the 15th International Conference on Artificial Intelligence and Law*, ICAIL 2015, pages 63–68, New York, NY, USA. ACM.

Hansen, J. (2014). Reasoning about permission and obligation. In S. O. Hansson, editor, *David Makinson on Classical Methods for Non-Classical Problems*, pages 287–333. Outstanding Contributions to Logic Volume 3, Springer.

Hobbs, J. and Gordon, A. (2017). *A formal theory of commonsense psychology, how people think people think*. Cambridge University Press.

LegalRuleML. (2019). *`https://www.oasis-open.org/committees/legalruleml/`*. OASIS consortium, `https://www.oasis-open.org`.

Makinson, D. and van der Torre, L. W. N. (2000). Input/output logics. *Journal of Philosophical Logic*, 29(4):383–408.

Makinson, D. and van der Torre, L. (2001). Constraints for input/output logics. *Journal of Philosophical Logic*, 30(2):155–185.

Malerba, A. (2017). *Interpretive Interactions among Legal Systems and Argumentation Schemes*. Ph.D. thesis, Joint International Doctoral (Ph.D.) Degree in Law, Science and Technology (LAST-JD).

McCarthy, J. (1980). Circumscription: A form of non-monotonic reasoning. *Artificial Intelligence*, (13):27–39.

MIREL. (2019). *MIning and REasoning with Legal texts*. `http://www.mirelproject.eu`.

Palmirani, M. and Vitali, F., (2011). *Akoma Ntoso for Legal Documents*, pages 75–100. Springer Netherlands, Dordrecht.

Palmirani, M., Martoni, M., Rossi, A., Bartolini, C., and Robaldo, L. (2018a). Legal ontology for modelling GDPR concepts and norms. In *The Thirty-first annual conference for Legal Knowledge and Information Systems (JURIX 2018)*, pages 91–100.

Palmirani, M., Martoni, M., Rossi, A., Bartolini, C., and Robaldo, L. (2018b). Pronto: Privacy ontology for legal compliance. In *Proc. of the 18th European Conference on Digital Government (ECDG 2018)*.

Pnueli, A. (1977). The temporal logic of programs. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, SFCS '77, pages 46–57. IEEE Computer Society.

Robaldo, L. and Sun, X. (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.

Robaldo, L., Caselli, T., Russo, I., and Grella, M. (2011). From italian text to timeml document via dependency parsing. In Alexander Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing*, pages 177–187, Berlin, Heidelberg. Springer Berlin Heidelberg.

Robaldo, L. and Bartolini, C. and Lenzini, G. (2019). *The DAPRECO knowledge base*. `https://github.com/dapreco/daprecokb/blob/master/gdpr/rioKB_GDPR.xml`.

Robaldo, L. (2010a). Independent set readings and generalized quantifiers. *The Journal of Philosophical Logic*, 39(1):23–58.

Robaldo, L. (2010b). Interpretation and inference with maximal referential terms. *The Journal of Computer and System Sciences*, 76(5):373–388.

Robaldo, L. (2011). Distributivity, collectivity, and cumulativity in terms of (in)dependence and maximality. *The Journal of Logic, Language, and Information*, 20(2):233–271.

Rotolo, A., Governatori, G., and Sartor, G. (2015). Deontic defeasible reasoning in legal interpretation: Two options for modelling interpretive arguments. In *Proceedings of the 15th International Conference on Artificial Intelligence and Law*, ICAIL 2015, New York, NY, USA. ACM.

Sartor, G. (2005). *Legal Reasoning: A Cognitive Approach to the Law*. Treatise of legal philosophy and general jurisprudence / ed.-in-chief Enrico Pattaro. Springer.

Searle, J. R. (1995). *The construction of social reality*. The Free Press, New York.

Sun, X. and Robaldo, L. (2017). On the complexity of input/output logic. *The Journal of Applied Logic, Vol. 25*.

Sun, X. and van der Torre, L. W. N. (2014). Combining constitutive and regulative norms in input/output logic. In Fabrizio Cariani, et al., editors, *Deontic Logic and Normative Systems - 12th International Conference, DEON 2014, Ghent, Belgium, July 12-15, 2014. Proceedings*, volume 8554 of *Lecture Notes in Computer Science*, pages 241–257. Springer.

Walton, D., Sartor, G., and Macagno, F. (2016). An argumentation framework for contested cases of statutory interpretation. *Artif. Intell. Law*, 24(1):51–91.