Formalizing GDPR Provisions in Reified I/O Logic: The DAPRECO Knowledge Base

Formalizing GDPR provisions in reified I/O logic: the DAPRECO knowledge base *

Livio Robaldo[§] Cesare Bartolini[§] Monica Palmirani[+]
Arianna Rossi[§] Michele Martoni[+] Gabriele Lenzini[§]

[§]University of Luxembourg,
Interdisciplinary Centre for Security, Reliability and Trust (SnT)
{*livio.robaldo,cesare.bartolini,arianna.rossi,gabriele.lenzini*}*@uni.lu*

[+]University of Bologna, CIRSFID
{*monica.palmirani,michele.martoni*}*@unibo.it*

Abstract

The DAPRECO knowledge base is the main outcome of the interdisciplinary project bearing the same name[1]. It is a repository of rules written in LegalRuleML, an XML formalism designed to be a standard for representing the semantic and logical content of legal documents. The rules represent the provisions of the General Data Protection Regulation (GDPR), the new Regulation that is significantly affecting the digital market in the European Union and beyond. The DAPRECO knowledge base builds upon the Privacy Ontology (PrOnto) (Palmirani et al., 2018c), 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 Input/Output logic (Robaldo and Sun, 2017) and then codified in LegalRuleML. Reified Input/Output logic is an application of standard Input/Output logic for legal reasoning, in which Input/Output logic is combined with the reification-based approach in (Hobbs and Gordon, 2017). The DAPRECO knowledge base is then a case study for reified Input/Output logic, and it shows that the formalism indeed appears to be a good candidate to effectively formalize, via uniform and simple (flat) representations, complex linguistic/deontic phenomena that may be found in legal texts. To date, the DAPRECO knowledge base is the biggest knowledge base in LegalRuleML and Input/Output logic freely available online[2].

[*] Research supported by the Luxembourg national FNR-CORE project “DAPRECO: DAta Protection REgulation Compliance”, and by the European Union’s Horizon 2020 research and innovation programme under the Marie Skodowska-Curie grant agreement No 690974 for the project “MIREL: MIning and REasoning with Legal texts”. Arianna Rossi performed this work at CIRSFID (University of Bologna) and ICR (University of Luxembourg) while she was supported by LAST-JD, the Joint International Doctoral Degree in Law, Science, and Technology, financed by EACEA.

[1] `https://www.fnr.lu/projects/data-protection-regulation-compliance`.

[2] `https://github.com/dapreco/daprecokb/blob/master/gdpr/rioKB_GDPR.xml`.

1 Introduction

The management of large repositories of norms, the semantic access to these norms, and the legal reasoning based on them are key challenges in Computational Law, the branch of Legal Informatics concerned with the computational analysis of norms occurring in legislation and related documents (jurisprudence, recommendations from authoritative sources, doctrine, etc.), which are originally available in natural language only.

Legal Informatics has recently received a lot of investments from industry and institutions, as witnessed by the EU projects MIREL[3] and LAST-JD[4], due to the well-known rise of RegTech and FinTech[5], which is in turn due to the 2008 global financial crisis (Arner, Barberis, and Buckey, 2016).

In the years 2010-2016, research in Computational Law had primarily focused on the application of Natural Language Processing (NLP) methods to legal texts, particularly to design 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., 2012; Boella et al., 2016).

Eunomos and similar systems classify, index, and discover inter-links between legal documents, retrieved through Web-crawling tools, by exploiting NLP tools, such as parsers and statistical algorithms, as well as semantic knowledge bases, such as legal ontologies in Web Ontology Language (OWL)[6]. This is often done by transforming the source legal documents into XML standards and tagging the relevant information. Subsequent phases are devoted to archiving and querying the XML files.

One of the most known legal XML standards is Akoma Ntoso[7] (Palmirani, 2011; Palmirani and Vitali, 2011), a.k.a. LegalDocML, a Committee specification by OASIS that defines a set of simple, technology-neutral representations of legislative and judiciary documents in XML format.

Akoma Ntoso provides in-line tags[8] to annotate, either manually or via entity-linking and concept-mining NLP techniques, textual information with respect to the classes and individuals of legal (computational) ontologies. 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 organized and connected to each other via basic semantic relations (is-a, part-of, etc.) that enable basic forms of reasoning, and they can 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 and retrieve information, its overall usefulness and effectiveness is limited due

[3] http://www.mirelproject.eu

[4] https://www.last-jd-rioe.eu

[5] FinTech (Financial Technology) refers to the use of artificial intelligence and computer science to support or enable banking and financial services. The main functions of FinTech software include regulatory monitoring, reporting, and compliance. RegTech (Regulatory Technology) is a more general term referring to computer technology applied to any kind of regulated business, not only finance.

[6] https://www.w3.org/OWL

[7] http://www.akomantoso.org

[8] See Section 5.5 of http://docs.oasis-open.org/legaldocml/akn-core/v1.0/akn-core-v1.0-part1-vocabulary.html.

to the focus on terminological issues and information retrieval, all the while disregarding the specific semantic aspects of law, in particular its logical structure in terms of constitutive and regulating rules, which are those that effectively allow legal reasoning: given a description of the state of affairs, logic rules determine what is obligatory and what is forbidden, which obligations have already been fulfilled, which ones have already been violated, which ones are still in force, as well as what the provisions allow. Such inferences provide precious information that can be expended, for instance, in decision making and risk assessment.

For this reason, recent research in Computational Law led to the identification of a new component devoted to *logic rules*, as exemplified in Figure 1.

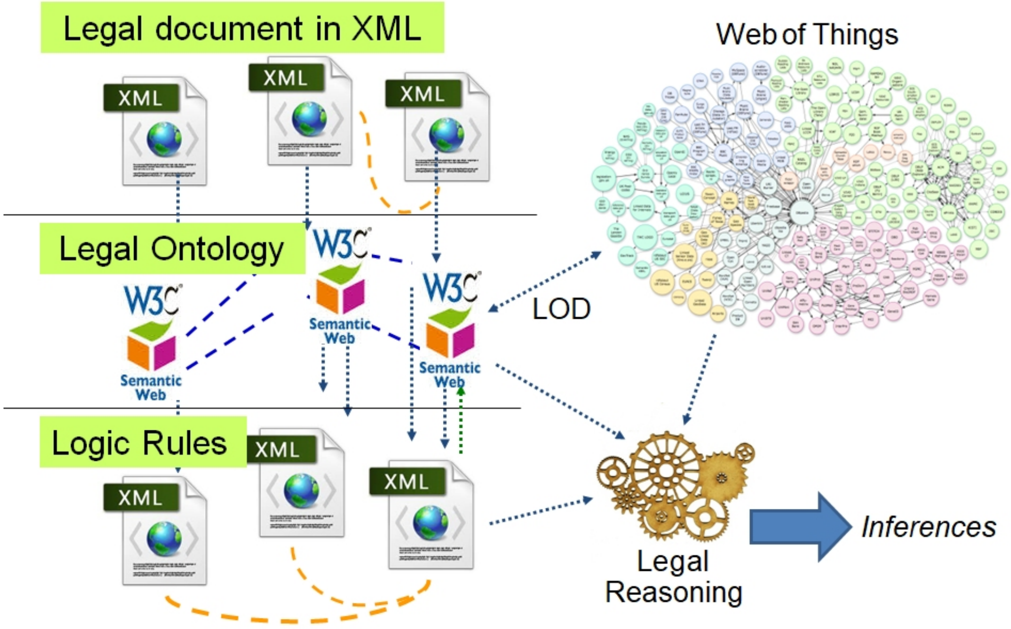


Figure 1: Three levels in Computational Law.

Legal document management systems such as Eunomos incorporate the first two levels in Figure 1, while the third one (logic rules) is mostly at the stage of basic research as of today. The research presented in this paper deals with this level.

A new standardization initiative called LegalRuleML[9] has been recently proposed by OASIS[10] to explicitly deal with the third level in Figure 1. LegalRuleML (Athan et al., 2013; Athan et al., 2015) is an XML format that extends the RuleML standard[11] to define a rule interchange language for the legal domain.

[9] https://www.oasis-open.org/committees/legalruleml/.

[10] https://www.oasis-open.org/.

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

While Akoma Ntoso is used to tag the original textual content of the legal documents, LegalRuleML separately represents and stores the logical content of the provisions. Specifically, LegalRuleML allows to specify semantic/logical representations in RuleML and associate them with both the structural elements (such as articles and paragraphs) of the Akoma Ntoso documents, as well as with the concepts in the legal ontology.

A very simple example is the LegalRuleML representation of the sentence “every man is obliged to run”, which may be simply represented by the following LegalRuleML rule:

```
<lrml:PrescriptiveStatement key=”someuniquekey”>
  <ruleml:Rule closure=”universal”>
    <ruleml:if>
      <ruleml:Atom>
        <ruleml:Rel iri=”man” />
        <ruleml:Var key=”:x”>x</ruleml:Var>
      </ruleml:Atom>
    </ruleml:if>
    <ruleml:then>
      <lrml:Obligation>
        <ruleml:Atom>
          <ruleml:Rel iri=”run” />
          <ruleml:Var keyref=”:x” />
        </ruleml:Atom>
      </lrml:Obligation>
    </ruleml:then>
  </ruleml:Rule>
</lrml:PrescriptiveStatement>
```

Predicates can be connected to ontological concepts, and the connection is encoded in LegalRuleML via the attribute `iri` of the tag `<ruleml:Rel>`. For instance, the LegalRuleML representation above involves the concepts “`man`” and “`run`”.

Note that LegalRuleML only provides a set of XML tags to encode formulæ in some logic. For instance, the logic of the previous formula could be taken as Standard Deontic Logic[12]. The formula is then standardly written as ‘$\forall_x[man(x) \rightarrow \mathbf{OB}(run(x))]$’. In the rest of the paper, all formulæ that we will encode in LegalRuleML are expressed in reified Input/Output logic (Robaldo and Sun, 2017), the representational language chosen for the research presented in this paper. We will extensively illustrate and discuss reified Input/Output logic and its features below.

This paper presents an implementation, formalized in reified Input/Output logic and encoded in LegalRuleML, of the third level in Figure 1, with respect to the General Data Protection Regulation (GDPR)[13]. Specifically, this paper presents the DAPRECO knowledge base[14], the main tangible output of the DAPRECO (DAta Protection REgulation COmpliance) research project[15] (Bartolini et al., 2016).

The GDPR brought about a small revolution in the world of online services, as attested by the fact that most service providers changed their privacy policies and introduced more fine-grained privacy controls for data subjects (Satariano, 2018). Enterprises dealing with personal data, including of course those operating in the RegTech and FinTech domains,

[12] `https://plato.stanford.edu/entries/logic-deontic/#2`

[13] Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC.

[14] `https://github.com/dapreco/daprecokb/blob/master/gdpr/rioKB_GDPR.xml`.

[15] `https://www.fnr.lu/projects/data-protection-regulation-compliance/`.

have become concerned with making their business compliant with the new legal regime. Failure to comply with the GDPR could imply major financial losses[16] and the inability to carry on their business, a fatal possibility in the frantically competing environment of Internet services.

As said above, in order to make a step forward in the basic research in Computational Law, the project DAPRECO pioneered the use of reified Input/Output logic and LegalRuleML to implement the third level in Figure 1. The GDPR has been selected as case study due to its impact to the nowadays society.

The DAPRECO knowledge is built upon recent research results, mainly developed at the University of Bologna, in the context of the MIREL (MIning and REasoning with Legal texts) project[17]. These results cover the first and the second level 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 representation of the GDPR are used within the DAPRECO knowledge base in order to associate the corresponding structural elements with the reified Input/Output logic formulæ representing their meaning.

- An ontology called PrOnto (Privacy Ontology) has been developed following a thorough ontology development methodology called MeLON (Palmirani et al., 2018d; Palmirani et al., 2018c; Palmirani et al., 2018b). The ontology has been then encoded in `OWL2-DL`[18]. The latest version of PrOnto is described in (Palmirani et al., 2018a). PrOnto is built according to a modular structure. The ontology is designed in such a way that it models the essential European data protection rules contained in the GDPR, but these rules can be easily extended by attaching additional ontologies (for example, for domain-specific provisions, or for Member State legislation). PrOnto concepts are also tagged within the Akoma Ntoso representation mentioned in the previous point and with the predicates used in the DAPRECO knowledge base, as shown below, via LegalRuleML tags.

The MeLOn methodology, on which the design of the PrOnto ontology has been grounded, follows standard principles of minimization, which may be found within main surveys on computational ontology design and evaluation (Brank, Grobelnik, and Mladenić, 2005; Bandeira et al., 2016). As a general rule in ontology engineering, design principles such as minimization and avoiding redundancy are needed to achieve computational efficiency (Bandeira et al., 2016).

However, PrOnto alone is not a suitable knowledge base truly fit for automatic legal reasoning. Simply put, a computational ontology in OWL only defines and describes the main concepts involved, as well as the main semantic relations between them, that may be useful to index information in the data protection domain, thus facilitating their navigation and search. This allows for basic reasoning, but it does not suffice to assess

[16] A particularly representative example is the “Délibération SAN-2019-001 du 21 janvier 2019”, issued by the Commission nationale de l’informatique et des libertés (CNIL), the Supervisory Authority of France, which sanctioned Google LLC for 50 million euros.

[17] `http://www.mirelproject.eu`

[18] `https://www.w3.org/TR/owl2-overview`

compliance checking. To this end, we also need fine-grained rules representing the obligations and permissions from the GDPR, as well as further concepts involved in these obligations and permissions.

In particular, it is well-known that, while assessing compliance with legislation, the matter of interpretation (that is, the fact that provisions are subject to different, and possibly incompatible, legal interpretations) needs to be taken into account. Furthermore, in the case of the GDPR, the Regulation itself requires the definition of additional documents (data protection policies, codes of conduct, etc., as further discussed in section 6) to fine-tune its provisions within specific contexts (Fintech, eHealth, IoT, etc.) or with respect to different technologies used for the processing. As technology advances, new official documents may be produced to override preexisting ones.

In order to account for this additional dimension, standard logical languages for legal reasoning are *defeasible*, a feature that is not implemented in PrOnto nor, more generally, in `OWL2-DL` (Casini et al., 2015).

Reified Input/Ouput logic has been precisely designed to represent complex, and possibly defeasible, deontic statements in natural language, by means of a simple formal machinery, thus facilitating the creation of large knowledge bases of machine-readable formulæ associated with existing legislative provisions.

The next sections describe the DAPRECO knowledge base while focusing on the Natural Language Semantics issues encountered during the development of the knowledge base, and how reified Input/Output logic is able to cope with them. In the future, we plan to build on top of the DAPRECO knowledge base towards a comprehensive reference knowledge base for applications in Legal Informatics in the data protection domain.

The rest of the paper is structured as follows: section 2 presents Input/Output logic as a general abstract formalism to model logical inferences while section 3 presents *reified* Input/Output logic, the underlying formalism of the DAPRECO knowledge base, which applies Input/Output logic for legal reasoning; the construction of the DAPRECO knowledge base, the issues encountered during its development, and their representation in reified Input/Output logic will be explained in section 4; on the other hand, section 5 provides an overview of related works, while comparing them with the solutions offered herein, and section 6, devoted to future works, discusses the necessary steps that we still need to make the DAPRECO knowledge base as truly usable in real-world applications; section 7 concludes the paper.

2 Background (1): Input/Output logic

Input/Output logic was introduced in (Makinson and van der Torre, 2000) as a general abstract formalism to model logical inferences, including certain non-standard ones in which “input propositions are not in general included among outputs, and the operation is not in any way reversible”.

Input/Output systems are a family of if-then rules in the form (a, b), such that when a is given in input, b is returned in output. The if-then rules are also called Input/Output pairs or Input/Output generators. Moreover, in the rest of the paper we will respectively term a and b as the LHS (left-hand side) and the RHS (right-hand side) of the Input/Output pair.

The LHSs and the RHSs of the Input/Output pairs are whole formulæ in some *object logic* with language L. In what follows, given a set A of formulæ in the object logic, $Cn(A)$ refers to the set of all formulæ derived from A through the inference rules of the logic; in symbols, $Cn(A) = \{a \in L\colon\ A \vdash a\}$.

Further axioms, a.k.a. derivation rules, may be then added to the Input/Output system in order to obtain different outputs, i.e., different *operational meanings*, when the system is fed with certain inputs.

For this reason, the semantics of Input/Output logic is said to be "norm-based" (Hansen, 2014). Other deontic frameworks based on norm-based semantics are imperative logic (Hansen, 2008), prioritized default logic (Horty, 2012), and defeasible deontic logic (Governatori et al., 2013). These frameworks represent an alternative to deontic frameworks based on *possible-worlds semantics*, e.g., STIT logic (Horty, 2001).

Norm-based semantics have been proposed as a straightforward solution to the well-known Jørgensen's dilemma (Jørgensen, 1937), saying that norms are entities different from declarative statements. Specifically, declarative statements may bear truth-values, which means they are capable of being true or false, while norms may be complied with or violated, but it makes no sense to state that they are true or false. Input/Output logic aims at solving Jørgensen's dilemma at its starting line, in that conditional norms do not bear truth values (see discussion in (Makinson and van der Torre, 2003b)).

In the past literature in Input/Output logic, the formalism was mostly used to model deontic and legal reasoning, and the same will be done in this paper. However, as said at the beginning of this section, Input/Output logic is a *general* formalism that may be likewise used to model other forms of reasoning; for instance (Bochman, 2004) uses it for modeling causal reasoning.

In (1) we report three axioms that we may impose on a set of pairs S belonging to an Input/Output system:

(1) - `SI` (strengthening the input): from (a, x) to (b, x), whenever $a \in Cn(\{b\})$

- `WO` (weakening the output): from (a, x) to (a, y), whenever $y \in Cn(\{x\})$

- `AND` (conjunction of output): from (a, x) and (a, y) to $(a, x \wedge y)$

Imposing, for instance, the axiom `SI` to a set of pairs S amounts to saying that, if S includes a pair $(a,\ x)$ and the formula a is derivable from the formula b through the derivation rules of the object logic, then also the pair $(b,\ x)$ *must* belong to S. The axiom `WO` imposes the same, but on the outputs rather than the inputs: if S includes a pair $(a,\ x)$, then S must also includes a pair $(a,\ y)$ for every formula y derivable from x through the derivations rules of the object logic. Finally, `AND` states that if S includes two pairs (a, x) and (a, y), then also the pair $(a, x \wedge y)$ (same LHS and RHS obtained by conjoining the RHSs of the two initial pairs) belongs to S.

By imposing the closure of `SI`, `WO`, and `AND` on a set of pairs S, we obtain an output configuration called "simple-minded output", denoted as out_1.

As said at the beginning of the section, "input propositions are not in general included among outputs". This is achieved only if we impose another axiom, called `ID`, that requires S to include the pair (a, a), for every formula a of the object logic:

(2) - `ID` (identity): from nothing to (a, a)

By imposing `ID` on out_1 we obtain its "throughput" version denoted as out_1^+. Two other axioms extensively studied in the Input/Output logic literature are `OR` and `CT`:

(3) - `OR` (disjunction of input): from (a, x) and (b, x) to $(a \vee b, x)$

- `CT` (cumulative transitivity): from (a, x) and $(a \wedge x, y)$ to (a, y)

`OR` states that if two inputs a and b produce the same output x, then the latter is produced also by their disjunction, i.e., $a \vee b$. By imposing `OR` on out_1 we obtain another configuration called "basic" and denoted as out_2. `OR` does not seem to be appropriate to model legal reasoning[19], so that we will no longer consider it below.

On the other hand, it is easy to see that the effect of `CT` on a set of pairs S is the one of reusing the outputs x as inputs again: by adding the pair (a, y) to S, we obtain the output y when the Input/Output system is fed with a; this is equivalent to feeding it with a and x, if S includes the pairs (a, x) and $(a \wedge x, y)$. For this reason, by imposing `CT` on out_1, we obtain a configuration called "reusable" and denoted as out_3. Again, by also imposing the axiom `ID` on out_3, we obtain its "throughput" version out_3^+.

(Parent and van der Torre, 2014) and (Parent and van der Torre, 2014) later showed that the combined effect of the axioms `CT` and `WO` could lead to certain paradoxes[20], so that they proposed a new version of Input/Output logic, called *aggregative* Input/Output logic, obtained by replacing `CT` and `WO` with the stronger axioms `ACT` and `EQ`:

(4) - `ACT` (aggregative cumulative transitivity): from (a, x) and $(a \wedge x, y)$ to $(a, x \wedge y)$

- `EQ`: (output equivalence) from (a, x) to (a, y), whenever $y \in Cn(\{x\})$ and $x \in Cn(\{y\})$

The literature about Input/Output logic has extensively shown that the formalism, in particular its aggregative version, is able to solve common paradoxes in Standard Deontic Logic. Two recent papers addressing the capabilities of Input/Output logic to solve these paradoxes are (Parent and van der Torre, 2017) and (Parent and van der Torre, 2018).

2.1 Using Input/Output logic for legal reasoning

The first relevant proposal to use Input/Output logic for legal reasoning has been (Boella and van der Torre, 2004b). While previous literature in Input/Output logic (Makinson and van der Torre, 2000; Makinson and van der Torre, 2001; Makinson and van der

[19] As discussed in (Robaldo and Sun, 2017), consider the obligations "If someone kills a dog, s/he has to spend two years in prison" and "If someone robs a bank s/he has to spend two years in prison". Suppose also that John did one of the two, but there is no way to come to know *which* one, i.e. whether he killed a dog or robbed a bank. Logically, John must spend two years in prison. But on the perspective of legal reasoning, he must not: only if concrete evidence of what he did is found, obligations apply. The example considered marks an interesting border between legal reasoning and standard logical reasoning.

[20] The example of paradox discussed in (Parent and van der Torre, 2014) is: given the obligations "You ought to exercise hard everyday" ($\top$, Ex) and "If you exercise hard everyday, you ought to eat heartily" (Ex, Eh), we derive via `CT` that also the obligation "You ought to eat heartily" ($\top$, Eh) holds. However, this is not clearly the case: you are obliged to eat heartily only if you exercise hard everyday.

Torre, 2003a) focused on regulative norms and deontic reasoning only (i.e., obligations and permissions), (Boella and van der Torre, 2004b) was the first attempt to also formalize constitutive norms, another kind of conditionals present in legal texts.

The distinction between regulative and constitutive norms is well-known in the literature (see (Searle, 1995), among others). Regulative norms specify what is obligatory or permitted in terms of certain abstract concepts called "institutional facts", whereas constitutive norms specify what *counts as* institutional facts in the context.

Formally, Input/Output systems for legal reasoning combine three sets of pairs:

(5) - O, which includes pairs (o_1, o_2) reading as "o_2 is obligatory given o_1".

- P, which includes pairs (p_1, p_2) reading as "p_2 is permitted given p_1".

- C, which includes pairs (c_1, c_2) reading as "c_1 counts as c_2".

The sets of Input/Output rules O, P, and C may be regulated by different output configurations. For instance, the definitions in (Boella and van der Torre, 2004b) use out_3 for obligations and permissions and out_3^+ for constitutive rules. In other words, both output configurations are "reusable", but only the one imposed on constitutive rules features a throughput from the input to the output, enforced by the axiom `ID`. As regulative rules may be directly applied to these inputs, we want them to also count as institutional facts. On the other hand, by imposing the axiom `ID` on the pairs in O (or in P), we would obtain that, for every institutional fact a, a is obligatory (or permitted) in the context, which is clearly not the case.

However, the formalization in (Boella and van der Torre, 2004b) does not consider the axioms `ACT` and `EQ`, which have been proposed only later on in (Parent and van der Torre, 2014). Moreover, it is indeed too complex for the purposes of our research: the framework in (Boella and van der Torre, 2004b) is grounded on the metaphor of normative systems as agents, in which regulative rules correspond to the goals of the agents and constitutive rules to their belief; in this settings, agents are assumed to be "obliged" to pursue their goals, and they can interact with each other by playing games.

Therefore, in the rest of the paper we will adhere to the simpler formalization proposed in (Sun and van der Torre, 2014). Specifically, the Input/Output framework in (Sun and van der Torre, 2014) may be depicted as two *sequential* Input/Output systems, in which the output of the first one is given in input to the second one; see Figure 2.

The first Input/Output system enforces constitutive norms (set of pairs C, under out_3^+, i.e., under axioms `SI`, `AND`, `ACT`, `EQ`, and `ID`) while the second one enforces obligations and permissions (set of pairs O and P, under out_3, i.e., under axioms `SI`, `AND`, `ACT`, `EQ`, and `ID`). Again, the difference between the two output operations is the axiom `ID`, which only holds on the constitutive if-then rules, so that the initial input facts are also given in input to the regulative if-then rules, i.e., they also "count as" institutional facts. On the other hand, the `ACT` axiom, holding in both Input/Output systems, makes the overall framework equivalent to one where the final output is "reused" as the initial input. See (Sun and van der Torre, 2014) for formal details.

The neat formal distinction between constitutive and regulative norms is particularly convenient when the normative system may be subject to revisions. This has been recently shown in (Maranhão, 2017), which extends Input/Output logic with constructs

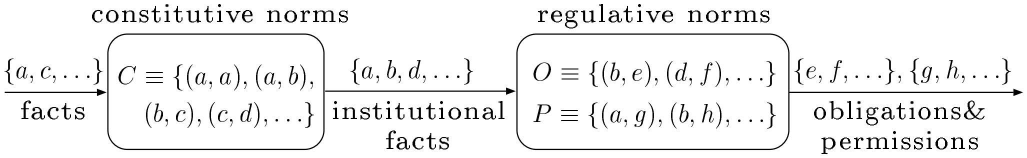

constitutive norms
regulative norms
$\{a, c, \ldots\}$
facts
$C \equiv \{(a, a), (a, b),$
$(b, c), (c, d), \ldots\}$
$\{a, b, d, \ldots\}$
institutional facts
$O \equiv \{(b, e), (d, f), \ldots\}$
$P \equiv \{(a, g), (b, h), \ldots\}$
$\{e, f, \ldots\}, \{g, h, \ldots\}$
obligations& permissions

Figure 2: An Input/Output framework for legal reasoning.

from the literature in belief revision, fit to revise the sets of Input/Output pairs against the introduction of new rules, abrogation of old ones, or ascription of legal interpretations to the rules. Maranhão and de Souza (2018) later formalizes three "contraction" operators for combined Input/Output systems, in order to update the sets of Input/Output pairs alongside the evolution of the legal framework modeled by these sets.

The Input/Output framework in Figure 2 is able to infer what is obligatory and permitted, given a description of the state of affairs (input facts).

To determine whether the regulative norms have been violated, it is sufficient to check whether the set of obligations in output contains a formula that is inconsistent with one of the input facts (see also subsection 5.2 below). For instance, let's assume that, in Figure 2, the input facts include $\neg e$. Then, the if-then rule $(b, e) \in O$ has been violated: e is obligatory with respect to the normative system and the input facts; but the latter also includes the negation of e.

Other reasoning tasks are desirable; for those tasks, it is necessary to introduce additional meta-structures to the Input/Output framework.

For instance, (Makinson and van der Torre, 2001) proposes to handle contrary-to-duty reasoning, i.e., the task of determining which obligations are operative in a situation that already violates some of them, by introducing a meta-structure $\texttt{outfamily}(O,\ A)$, where A is the input of the pairs in O (note that, with respect to Figure 2, A is the set of institutional facts). Taken $out_3(O,\ A)$ as the set of RHSs in O whose LHSs occur in A, $\texttt{outfamily}(O,\ A)$ is defined as the family of all $out_3(O',\ A)$ such that $O' \subseteq O$ and A is consistent with $out_3(O',\ A)$. The reader interested in the (general) formal definitions and proofs is addressed to (Makinson and van der Torre, 2001). Here we just report a simple example to understand how $\texttt{outfamily}(O,\ A)$ works. Suppose we have two obligatory norms: "The cottage must not have a fence or a dog" (in symbols: $(\top,\ \neg(f \vee d)) \in O$) and "If the cottage has a dog it must have both a fence and a warning sign." (in symbols: $(d,\ f \wedge w) \in O$). Suppose further that we are in the situation where the cottage has a dog (in symbols: $d \in A$). In this context, the first norm is violated, as d is inconsistent with $\neg(f \vee d)$. Still, we would like to infer that we are still obliged to build a fence with a warning sign around the cottage. Assuming O is restricted by the axiom $\texttt{EQ}$, as in Figure 2, $\texttt{outfamily}(O,\ A)$ precisely enables that inference, in that it contains the targeted obligation: $\texttt{outfamily}(O,\ A) \equiv \{\{s :\ s \leftrightarrow (f \wedge w)\}\}$.

Other similar meta-structures have been defined in (Makinson and van der Torre, 2003a) to identify and reason with different kinds of permissions. The if-then rules in P are called positive static permissions, as they explicitly state what is permitted in the normative system. More generally, positive permissions are those that may be deductively derived from P and O.

However, from a philosophical perspective, it may be argued that all that is not forbidden is permitted, even if it is not explicitly asserted. This is another notion of permission, called negative permission.

A much more subtle notion of permission, called positive dynamic permission, is identified as follows: a formula x is a positive dynamic permission in a certain state of affairs A if and only if the prohibition of x given A (in symbols: $(\bigwedge A, \neg x) \in O$) creates inconsistency in the normative system for some possible sets of input facts. In other words, these permissions hold if and only if prohibiting them would produce inconsistencies for the normative system with respect to any state of affairs that can be given in input. As (Makinson and van der Torre, 2003a) states, dynamic positive permissions behave like "amplified" negative permission.

The formal definitions and proofs of the meta-structures needed to reason on this kind of permissions are quite complex and beyond the scope of the present paper; the interested reader is addressed to (Makinson and van der Torre, 2003a).

Further reasoning tasks worthy of consideration concern moral conflicts. In fact, violating the norms is not always the worst possible action agents can do. Thus, agents should be indeed allowed to violate them, while evaluating the penalties and the risks of these violations compared to the nonfulfillment of the corresponding goals. For instance, suppose that I do not have the money for buying the ticket for the bus but I have to rush to the hospital as I was just informed that my mother has been brought there. In such a case, the best action to do could arguably be to take the bus and risk the fine. Parent (2011) proposes to deal with moral conflicts in terms of additional meta-structures based on priorities. On the other hand, (Sun and Robaldo, 2015) define different kinds of agents (moral, amoral, negatively impartial, and positively impartial) with respect to a normative system formalized in Input/Output logic.

Computational complexity results for basic satisfiability tasks in Input/Output logic, including contrary-to-duty reasoning and the three notions of permissions mentioned above, have been presented in (Sun and Robaldo, 2017).

Specifically, it has been shown that Input/Output logic is decidable, although intractable, contrary to main deontic frameworks based on possible-worlds semantics; for instance, it has been proved in (Schwarzentruber and Semmling, 2014) that STIT logic is undecidable. The reason is that possible-worlds semantics adds an extra machinery that compromises decidability. On the contrary, the "if-then" architecture of the Input/Output logic pairs does not require such an extra effort while computing the aforementioned reasoning tasks.

On the other hand, (Sun and Robaldo, 2017) studied the complexity of Input/Output logic while considering standard propositional logic as the object logic. Since this is not the object logic used in reified Input/Output logic, as shown in the next subsection, the results in (Sun and Robaldo, 2017) do not hold here. We will come back to this issue again in subsection 6.2 below, devoted to future works.

3 Background (2): *reified* Input/Output logic

Reified Input/Output logic (Robaldo and Sun, 2017) is a recent formalism for representing and reasoning on norms originally available in natural language. It has been chosen in the project DAPRECO as the underlying formalism to represent GDPR provisions.

Reified Input/Output logic is an *application* (and not an extension) of Input/Output logic for Natural Language Semantics, designed for legal reasoning.

As it should be clear from the previous section, Input/Output logic is an *abstract* framework to represent and detach conditionals, i.e., a meta-logic to unwrap formulæ written in another language, called the object logic, when the system is fed with formulæ in the same language. Input/Output logic can be used to model *any* kind of reasoning, not only deontic reasoning.

We can then obtain *specialized* Input/Output systems by choosing: (1) different combinations of axioms, each of which defines a different relation between the outputs and the inputs ($out_1^{\{+\}}$, $out_2^{\{+\}}$, $out_3^{\{+\}}$, etc.), and (2) different object logics.

Concerning (1), following (Sun and van der Torre, 2014) and (Boella and van der Torre, 2004b), in reified Input/Output logic we use two sequential Input/Output systems, one for constituency rules and one for regulative rules, as depicted in Figure 2; we use out_3^+ for the first system and out_3 for the second one.

On the other hand, the object logic is the logic proposed in (Hobbs and Gordon, 2017), described in the next subsection. In past Input/Output logic literature, the object logic has been mostly standard propositional logic. However, as it is well-known, the expressivity of standard propositional logic does not suffice to adequately represent natural language utterances in practical applications. On the contrary, (Hobbs and Gordon, 2017) includes First Order Logic (FOL) terms to achieve the required expressivity.

3.1 The neo-Davidsonian approach in (Hobbs and Gordon, 2017)

The framework in (Hobbs and Gordon, 2017)[21] is a wide-coverage logic for Natural Language Semantics able to handle a fairly large set of linguistic phenomena into a simple logical formalism.

It is grounded on the notion of *reification*, a concept originally introduced in (Davidson, 1967). Modern logical approaches based on reification, such as (Hobbs and Gordon, 2017), are known in the literature as 'neo-Davidsonian' approaches.

Reification allows complex natural language statements to be expressed in FOL, by formalizing them such that events, states, and the like correspond to FOL constants or variables. Every predication in FOL, 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", from (Bach, 1981). The term e_b is the reification[22] of John's "blond-ness", i.e., it represents *the fact that* John is blond.

Other properties or events may be then applied to e_b, and recursively reified into new eventualities. For instance, we may represent the sentence "John wishes to be blond" with a predicate *wish* which takes `John` and e_b as arguments; then, John's wishing action may be reified into a new eventuality e_w:

[21] See also the manuscripts at `http://www.isi.edu/~hobbs/csk.html` and `http://www.isi.edu/~hobbs/csknowledge-references/csknowledge-references.html` for a quick introduction to the logical framework.

[22] States, facts, and events are *reified* into FOL individuals, from the Latin word "re(s)" for "thing": we take states, facts, and events to be *things*.

(6) $(wish'\, e_w\ \texttt{John}\ e_b) \wedge (blond'\, e_b\ \texttt{John})$

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 the predication '$(RexistAtTime\ e_w\ t)$'.

Only eventualities for which $RexistAtTime$ is asserted on t really exist at time t. $RexistAtTime$ resembles closely the predicate $HoldsAt$, used in Event Calculus approaches (see (Galton, 2006)).

The final representation[23] of the sentence "John wishes to be blond" (at time t) is:

(7) $(RexistAtTime\ e_w\ t) \wedge (wish'\ e_w\ \texttt{John}\ e_b) \wedge (blond'\ e_b\ \texttt{John})$

The explicit representation of time in FOL terms, such as "t", is a mandatory requirement for machine-readable representations in the legal domain. It is well-known (see (Ajani et al., 2017), Section 4.4) that overridden legal concepts still apply retroactively in the time span where they were in force. For instance, the norms in the GDPR are valid only from the date in which the Regulation states that those norms became applicable (25 May 2018). For infringements of the protection of personal data that occurred before that date, Directive 95/46/EC (or, rather, its Member State implementations, since a Directive is not applicable *per se*, but it needs to be enacted in Member State law) still applies. Therefore, time needs to be represented, in that legal reasoning tasks must only consider norms that hold in specific intervals of time. For instance, for all formulæ in the DAPRECO knowledge base and for all FOL terms t referring to instants of time, it holds "$t \geq$ 25 May 2018".

The logical framework in (Hobbs and Gordon, 2017) is characterized by a massive use of reification. Every relation on eventualities, including boolean operators, causal and temporal relations, and even tense and aspect, may be reified into another eventuality.

For instance, in (Hobbs and Gordon, 2017), '$(not'\ e_1\ e_2)$' is used to assert that e_1 is the eventuality of e_2's not existing, while the predication '$(or'\ e\ e_1\ e_2)$' 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. For instance, we may model their meaning via the axioms in (8) and (9) respectively. The former states that, for every instant t, 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. The latter states that, for every instant t, 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 as well.

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

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

[23] Note that in the formulæ, all terms are FOL constants. They could be FOL variables, provided that we quantify them, e.g., '$\exists_{e_b}[(blond'\ e_b\ \texttt{John})]'$.

What is important to understand is that not' and or' are not boolean operators. They are FOL predicates relating two and three eventualities respectively. The axioms in (8) and (9) define their meaning in terms of the boolean operators '$\neg$' and '$\vee$'.

In reified Input/Output logic, we distinguish between formulæ belonging to the assertive contextual statements (ABox), such as (6) and (7), from formulæ belonging to the terminological declarative statements (TBox), such as (8) and (9).

The former are flat conjunctions of atomic predications, and the only possible inferences are those allowed by conjunction ('$A, B \vdash A \wedge B$' and '$A \wedge B \vdash A$').

On the other hand, the latter may be any formula in standard FOL, possibly including other boolean operators besides '$\wedge$'.

With respect to Figure 2, the ABox is represented by the sets of pairs C, O, and P, while the TBox corresponds to the derivation rules allowed on the predications occurring in these pairs. The application of these derivation rules has been termed as "$Cn(A)$" at the beginning of the previous section, where A is a set of input formulæ; we remind that $Cn(A)$ is involved in the definitions of the axioms `SI`, `WO`, and `EQ`.

Thus, if A is a conjunction of the predications in (Hobbs and Gordon, 2017) that constitutes either the LHS or the RHS of an Input/Output if-then rule, $Cn(A)$ is the set of all formulæ derived from A through the statements in the TBox. In the case of the DAPRECO knowledge base, the TBox includes the semantic relations codified in PrOnto, e.g., the is-a relations between the ontological classes, as well as definitions of other needed predicates, such as (8) and (9) above for negation and disjunction[24].

Therefore, as explained in (Robaldo and Sun, 2017), in reified Input/Output logic the complexity is fully moved to the TBox, in that the ABox only includes conjunctions of atomic FOL sentences, and their complexity is then trivial. This feature appears pivotal for building and keeping under control large knowledge bases of formulæ, which was one of the main motivations behind the design of reified Input/Output logic.

In practical applications, the size of the TBox is usually much lower than the size of the ABox. Therefore, legal practitioners, who usually have little expertise in logic, could actively collaborate in the building of (large) ABoxes of formulæ representing norms from existing legislation. On the other hand, the construction of the TBox, i.e., the construction of the reference legal ontology, is left to researchers having advanced expertise in knowledge representation and ontology modeling. We will further elaborate on the distinction between ABox and TBox in subsection 6.2 below, devoted to future works.

For the time being, it should be clear that, thanks to reification, we are able to avoid nestings of subformulæ within complex operators, for the formulæ in the ABox. This is the main insight of the approach in (Hobbs and Gordon, 2017): using solely the simple formal mechanism of reification to model complex operators for causality, time, space, and the like, as well as for different modalities.

In light of this, combining (Hobbs and Gordon, 2017) and Input/Output logic appears to be a promising choice. Formally, Input/Output logic is a *flat* set of if-then rules, constrained by separate axioms, aiming at modeling the same meaning modeled as standard deontic logic, without the complex deontic operators that embed (nested) subformulæ.

[24] In the DAPRECO knowledge base, special LegalRuleML prefixes allow to distinguish the predicates corresponding to concepts in PrOnto from the others. See section 4 below.

3.2 Combining the approach in (Hobbs and Gordon, 2017) and Input/Output logic

(Hobbs and Gordon, 2017) is the object logic of reified Input/Output logic, meaning that the LHSs and the RHSs of the pairs in O, P, and C are formulæ in (Hobbs and Gordon, 2017). The output operations out_3^+ and out_3 enforce and constrain the detachment of the RHSs when the Input/Output systems are fed with the LHSs, as explained above.

A preliminary example is the obligation in (10), which is formalized with the reified Input/Output logic formula in (11).

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

(11) $\forall_x \forall_t ($

$$\exists_{e_o}\exists_{e_n}\exists_{e_b}\exists_{e_w}\exists_{t_1}[(RexistAtTime\ e_o\ t) \wedge (or'\ e_o\ e_n\ e_b) \wedge (not'\ e_n\ e_w) \wedge (wearing'\ e_w\ \ x\ \ t_1) \wedge (tie\ \ t_1) \wedge (blond'\ e_b\ \ x)],$$

$$\exists_{e_l}[(RexistAtTime\ e_l\ t) \wedge (leave'\ e_l\ \ x\ \ \texttt{Room})]\ ) \in O$$

In (11), 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 LHS to the RHS. Only variables which occur in both the LHS and the RHS are bound by these quantifiers. The formulæ in (11) also include existential quantifiers, which outscope the LHSs and RHSs and, as explained in (Robaldo and Sun, 2017), may be safely removed via Skolemization.

The formula in (11) reads as follows: for every individual x and for every time t, if it really exists that either x does not wear a tie t_1 or x is blond, then the real existence of a "leaving" action from the room, performed by x, is obligatory.

On the other hand, we may introduce the constitutive rule in (12) in order to state that whoever wears a tie is considered ("counts as") elegant.

(12) $\forall_x \forall_t (\ \exists_{e_w}\exists_{t_1}[(RexistAtTime\ e_w\ t) \wedge (wearing'\ e_w\ \ x\ \ t_1) \wedge (tie\ \ t_1)],$

$$\exists_{e_e}[(RexistAtTime\ e_e\ t) \wedge (elegant'\ e_e\ \ x)]\ ) \in C$$

A more complex example, taken from the DAPRECO knowledge base, is the representation in reified Input/Output logic of the provision in Article 12, paragraph 7, of the GDPR, shown in (13). Other examples are provided in (Bartolini et al., 2016; Robaldo and Sun, 2017).

(13) `GDPR (Art.12, par.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.

The provision in (13) contains both a permission and an obligation. In the DAPRECO knowledge base, these are expressed using the formulæ in (14) and (15), respectively.

$$
\begin{array}{ll}
(14) & \forall_t \forall_y \forall_{e_n} (\\
& \quad \exists_{a_1} \exists_{e_p} \exists_{e_{dp}} \exists_w \exists_z \exists_x \exists_i [\ (\mathit{RexistAtTime}\ a_1\ t) \wedge (\mathit{and}'\ a_1\ e_p\ e_n\ e_{dp}) \wedge \\
& \qquad (\mathit{DataSubject}\ w) \wedge (\mathit{PersonalData}\ z\ w) \wedge (\mathit{Controller}\ y\ z) \wedge \\
& \qquad (\mathit{Processor}\ x) \wedge (\mathit{nominates}\text{'}\ e_{dp}\ y\ x) \wedge \\
& \qquad (\mathit{PersonalDataProcessing}\text{'}\ e_p\ x\ z) \wedge (\mathit{Communicate}\text{'}\ e_n\ y\ w\ i)], \\
& \quad \exists_{eat} \exists_{ic} [(\mathit{RexistAtTime}\ e_{at}\ t) \wedge (\mathit{AttachTo}'\ e_{at}\ y\ ic\ \ e_n) \wedge (\mathit{Icon}\ ic)]\ ) \in P
\end{array}
$$

The formula in (14) contains an *and′* predicate[25]. This is a relation between multiple eventualities: its first argument (a_1, in (14)) really exists if, and only if, all other arguments (e_p, e_n, and e_{dp}, in (14)) really exist. All other predicates in (14) parallel NL words and do not need particular explanations.

The formula reads as follows: whenever 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 the notification e_n.

As said above, the provision in (13) also contains an obligation: in case the icon is in electronic form, it ought to be machine-readable. This is represented by the reified Input/Output logic formula in (15):

$$
\begin{array}{ll}
(15) & \forall_{t_1} \forall_{ic} (\\
& \quad \exists_{a_1} \exists_{e_p} \exists_{e_n} \exists_{e_{at}} \exists_{e_l} \exists_{e_{dp}} \exists_w \exists_z \exists_y \exists_x \exists_i [(\mathit{RexistAtTime}\ a_1\ t_1) \wedge \\
& \qquad (\mathit{and}'\ a_1\ e_p\ e_n\ e_{at}\ e_l\ e_{dp}) \wedge (\mathit{DataSubject}\ w) \wedge (\mathit{PersonalData}\ z\ w) \wedge \\
& \qquad (\mathit{Controller}\ y\ z) \wedge (\mathit{Processor}\ x) \wedge (\mathit{nominates}\text{'}\ e_{dp}\ y\ x) \wedge \\
& \qquad (\mathit{PersonalDataProcessing}\text{'}\ e_p\ x\ z) \wedge (\mathit{Communicate}\text{'}\ e_n\ y\ w\ i) \wedge \\
& \qquad (\mathit{AttachTo}'\ e_{at}\ y\ ic\ \ e_n) \wedge (\mathit{Icon}\ ic) \wedge (\mathit{electronicForm}'\ e_l\ ic)], \\
& \quad \exists_{e_{mr}} [(\mathit{RexistAtTime}\ e_{mr}\ t_1) \wedge (\mathit{machineReadableness}'\ e_{mr}\ ic)]\ ) \in O
\end{array}
$$

A significant difference between (15) and (14) is that, in (15), the LHS also requires the real existence of e_{at} and of e_l. The former models the action of attaching an icon to the notification, performed by the controller y, while the latter represents *the fact that* the icon ic is in electronic form. In case these eventualities really exist as well, then the "machine-readable-ness" of ic is required, or, in other words, *the fact that* the icon is machine-readable must obligatorily exist.

4 Building the DAPRECO knowledge base

The current version of the DAPRECO knowledge base includes 966 formulæ in reified Input/Output logic: 271 obligations, 76 permissions, and 619 constitutive rules. The number of constitutive rules is much higher than the one of obligations and permissions

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

as the knowledge base includes constitutive rules needed to trigger special inferences, as explained below in subsection 4.1.

The formulæ refer to the paragraphs or sub-paragraphs in the GDPR. It is possible to associate more than one formula with a single paragraph. This is the case, for instance, of Article 12, paragraph 7, shown above in (13), which is associated both with an obligation and with a permission.

Not all articles of the GDPR are currently covered. In particular, the DAPRECO knowledge base is conceived for being used within future applications, mainly for compliance checking. For this reason, its main target is the set of stakeholders that are involved in the actual personal data processing activities, such as controllers and processors. Conversely, we are not currently aiming at monitoring GDPR compliance with respect to the organization of supervisory authorities and administrative structures. We therefore skip the formalization of the provisions in chapters VI and VII, specifically from Article 51 to Article 76, which state the duties and powers of the data protection supervisory authorities and the European Data Protection Board. In future works, the DAPRECO knowledge base will be possibly extended to include such provisions.

The reified Input/Output logic formulæ in the DAPRECO knowledge base are formalized in LegalRuleML. Indeed, a very limited set of LegalRuleML tags are used to this end, in that, as explained in the previous section, reification allows to deal with complex semantic phenomena in terms of formally simple logical constructs.

As pointed out in section 1, LegalRuleML allows for a threefold interconnection between the structural items in the legal document (paragraphs, subparagraphs, and other structural components), the concepts in the PrOnto ontology, and the deontic formulæ expressing the meaning of the spans of text.

In LegalRuleML, legal sources may be referred via the `LegalReference` tag, as shown in the following example:

```
<lrml:LegalReference refersTo=”gdprC3S1A12P7ref” refID=”GDPR:art_12__para_7” />
```

where “art_12__para_7” is the `eId` of Article 12, paragraph 7, in the Akoma Ntoso file of the GDPR:

```
<paragraph eId=”art_12__para_7”>
  <num>7.</num>
  <content>
    <p>The information to be ... they shall be machine-readable.</p>
  </content>
</paragraph>
```

Legal sources may be associated with statements, i.e., collections of semantic representations, such as reified Input/Ouput formulæ, via the LegalRuleML tag `Association`:

```
<lrml:Association>
  <lrml:appliesSource keyref=”#gdprC3S1A12P7ref” />
  <lrml:toTarget keyref=”#statements38” />
</lrml:Association>
```

Finally, statements are formalized via the LegalRuleML tag `Statements`. Each `Statements` tag contains one or more `ConstitutiveStatement` tags, each corresponding to an Input/Output if-then rule. To distinguish which of the sets O, P, or C of the Input/Output normative system in Figure 2 the if-then rules belongs to, we use the LegalRuleML tag `Context`. For instance, in order to assert that the first formula of the `Statements` with `key=statements38`, shown below in Listing 1, is a permission, the DAPRECO knowledge base includes the following:

```
<lrml:Context key=”context_2” type=”rioOnto:permissionRule”>
  ...
  <lrml:inScope keyref=”#statements38Formula1” />
  ...
</lrml:Context>
```

The (compressed) LegalRuleML representation of the formulæ (14) and (15) is reported in Listing 1; the full version is online[26].

Listing 1: LegalRuleML representation of formulæ (14) and (15).

```
<lrml:Statements key=”statements38”>
  <lrml:ConstitutiveStatement key=”statements38Formula1”>
    <ruleml:Rule closure=”universal”>
      <ruleml:if>
        <ruleml:Exists>
          <ruleml:Var key=”:a1”>a1</ruleml:Var>
           ...
          <ruleml:Var key=”:i”>i</ruleml:Var>
          <ruleml:And>
            <ruleml:Atom>
              <ruleml:Rel iri=”rioOnto:RexistAtTime” />
              <ruleml:Var keyref=”:a1” />
              <ruleml:Var key=”:t1”>t1</ruleml:Var>
            </ruleml:Atom>
             ...
            <ruleml:Atom keyref=”:A620”>
              <ruleml:Rel iri=”prOnto:Communicate” />
              <ruleml:Var keyref=”:en” />
              <ruleml:Var keyref=”:y” />
              <ruleml:Var keyref=”:w” />
              <ruleml:Var keyref=”:i” />
            </ruleml:Atom>
          </ruleml:And>
        </ruleml:Exists>
      </ruleml:if>
      <ruleml:then>
        <ruleml:Exists>
          <ruleml:Var key=”:eat”>eat</ruleml:Var>
          <ruleml:Var key=”:ic”>ic</ruleml:Var>
          <ruleml:And>
            <ruleml:Atom>
              <ruleml:Rel iri=”rioOnto:RexistAtTime” />
              <ruleml:Var keyref=”:eat” />
              <ruleml:Var keyref=”:t1” />
            </ruleml:Atom>
            <ruleml:Atom keyref=”:A621”>
              <ruleml:Rel iri=”dapreco:AttachTo” />
              <ruleml:Var keyref=”:eat” />
              <ruleml:Var keyref=”:y” />
              <ruleml:Var keyref=”:ic” />
              <ruleml:Var keyref=”:en” />
            </ruleml:Atom>
            <ruleml:Atom>
              <ruleml:Rel iri=”dapreco:Icon” />
              <ruleml:Var keyref=”:ic” />
```

[26] https://github.com/dapreco/daprecokb/blob/master/gdpr/rioKB_GDPR.xml

```
            </ruleml:Atom>
          </ruleml:And>
        </ruleml:Exists>
      </ruleml:then>
    </ruleml:Rule>
  </lrml:ConstitutiveStatement>

  <lrml:ConstitutiveStatement key=”statements38Formula2”>
    <ruleml:Rule closure=”universal”>
      <ruleml:if>
        <ruleml:Exists>
          <ruleml:Var key=”:a1”>a1</ruleml:Var>
          ...
          <ruleml:Var key=”:i”>i</ruleml:Var>
          <ruleml:And>
            <ruleml:Atom>
              <ruleml:Rel iri=”rioOnto:RexistAtTime” />
              <ruleml:Var keyref=”:a1” />
              <ruleml:Var key=”:t1”>t1</ruleml:Var>
            </ruleml:Atom>
            ...
            <ruleml:Atom keyref=”:A625”>
              <ruleml:Rel iri=”dapreco:AttachTo” />
              <ruleml:Var keyref=”:eat” />
              <ruleml:Var keyref=”:y” />
              <ruleml:Var key=”:ic”>ic</ruleml:Var>
              <ruleml:Var keyref=”:en” />
            </ruleml:Atom>
            <ruleml:Atom>
              <ruleml:Rel iri=”dapreco:Icon” />
              <ruleml:Var keyref=”:ic” />
            </ruleml:Atom>
            <ruleml:Atom keyref=”:A626”>
              <ruleml:Rel iri=”dapreco:electronicForm” />
              <ruleml:Var keyref=”:el” />
              <ruleml:Var keyref=”:ic” />
            </ruleml:Atom>
          </ruleml:And>
        </ruleml:Exists>
      </ruleml:if>
      <ruleml:then>
        <ruleml:Exists>
          <ruleml:Var key=”:emr”>emr</ruleml:Var>
          <ruleml:And>
            <ruleml:Atom>
              <ruleml:Rel iri=”rioOnto:RexistAtTime” />
              <ruleml:Var keyref=”:emr” />
              <ruleml:Var keyref=”:t1” />
            </ruleml:Atom>
            <ruleml:Atom keyref=”:A627”>
              <ruleml:Rel iri=”dapreco:machineReadableness” />
              <ruleml:Var keyref=”:emr” />
              <ruleml:Var keyref=”:ic” />
            </ruleml:Atom>
          </ruleml:And>
        </ruleml:Exists>
      </ruleml:then>
    </ruleml:Rule>
  </lrml:ConstitutiveStatement>
</lrml:Statements>
```

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 the GDPR, Article 5, paragraph 1, point 1, 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 the PERSONALDATAPROCESSING class. The attribute is true when the processing is lawful, and false otherwise.

However, the PrOnto ontology does not specify the conditions under which the LAWFULNESS attribute is set to either true or false. This is achieved by means of a set of formulæ in the DAPRECO knowledge base.

For instance, Article 6, paragraph 1, point 1, of the GDPR specifies that one of the possible conditions under which the processing is lawful if that "the data subject has given consent to the processing of his or her personal data for one or more specific purposes;".

This is formalized via the reified Input/Output logic formula[27] in (16), which is in turn codified in the LegalRuleML constitutive statement shown in Listing 2.

$$\begin{aligned}
(16)\ \forall_{e_p}(\ &\exists_{a_1}\exists_{t_1}\exists_{e_{hc}}\exists_{e_{au}}\exists_{e_{dp}}\exists_w\exists_z\exists_y\exists_x\exists_c[\ (RexistAtTime\ a_1\ t_1) \wedge \\
&\qquad (and'\ a_1\ e_p\ e_{hc}\ e_{au}\ e_{dp}\ ) \wedge (DataSubject\ w) \wedge (PersonalData\ z\ w) \wedge \\
&\qquad (Controller\ y\ z) \wedge (Processor\ x) \wedge (nominates\text{'}\ e_{dp}\ y\ x) \wedge \\
&\qquad (PersonalDataProcessing\text{'}\ e_p\ x\ z) \wedge (Purpose\ e_{pu}) \wedge (Consent\ c) \wedge \\
&\qquad (isBasedOn\ e_p\ e_{pu}) \wedge (GiveConsent'\ e_{hc}\ w\ c) \wedge \\
&\qquad (AuthorizedBy'\ e_{au}\ e_{pu}\ c)\ ], \\
&(lawfulness\ e_p)\ ) \in C
\end{aligned}$$

Listing 2: LegalRuleML representation of formula 16.

```
<lrml:ConstitutiveStatement key="statements7Formula1">
  <ruleml:Rule closure="universal">
    <ruleml:if>
      <ruleml:Exists>
        <ruleml:Var key=":a1">a1</ruleml:Var>
        ...
        <ruleml:Var key=":c">c</ruleml:Var>
        <ruleml:And>
          <ruleml:Atom>
            <ruleml:Rel iri="rioOnto:RexistAtTime" />
            <ruleml:Var keyref=":a1" />
            <ruleml:Var keyref=":t1" />
          </ruleml:Atom>
          <ruleml:Atom>
            <ruleml:Rel iri="rioOnto:and" />
            <ruleml:Var keyref=":a1" />
            <ruleml:Var key=":ep">ep</ruleml:Var>
            <ruleml:Var keyref=":ehc" />
            <ruleml:Var keyref=":eau" />
            <ruleml:Var keyref=":edp" />
          </ruleml:Atom>
          <ruleml:Atom>
            <ruleml:Rel iri="prOnto:DataSubject" />
            <ruleml:Var keyref=":w" />
          </ruleml:Atom>
          <ruleml:Atom>
            <ruleml:Rel iri="prOnto:PersonalData" />
```

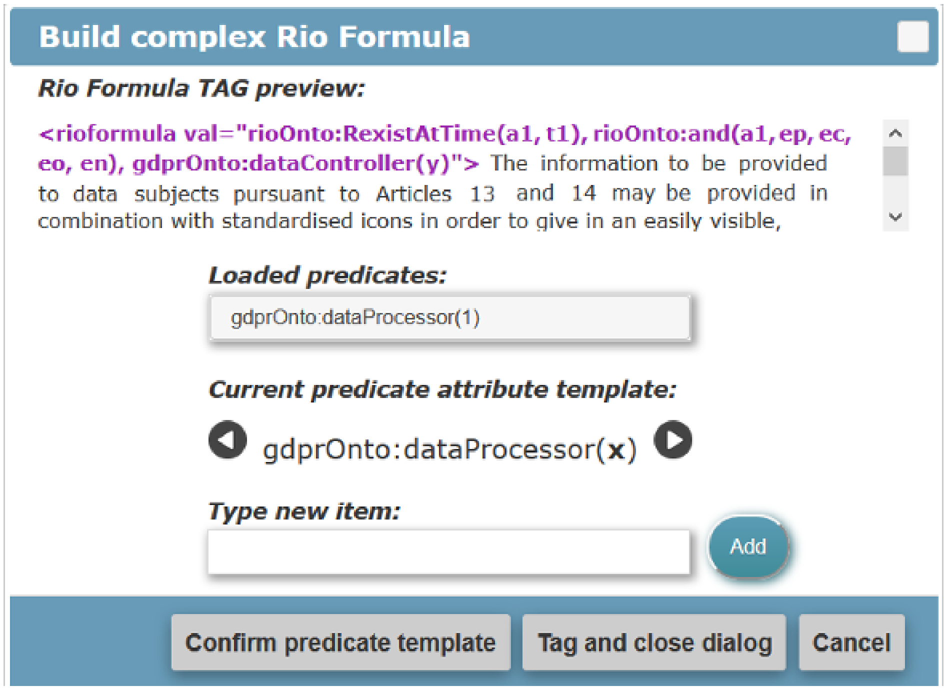
[27] Indeed, the formula in (16) lacks a predicate referring to an exception to Article 6, paragraph 1, point 1. The handling of exceptions in reified Input/Output logic will be illustrated below in subsection 4.2, so that we avoid that predicate in (16).

```
            <ruleml:Var keyref=”:z” />
            <ruleml:Var keyref=”:w” />
          </ruleml:Atom>
          <ruleml:Atom>
            <ruleml:Rel iri=”prOnto:Controller” />
            <ruleml:Var keyref=”:y” />
            <ruleml:Var keyref=”:z” />
          </ruleml:Atom>
          <ruleml:Atom>
            <ruleml:Rel iri=”prOnto:Processor” />
            <ruleml:Var keyref=”:x” />
          </ruleml:Atom>
          <ruleml:Atom keyref=”:A120”>
            <ruleml:Rel iri=”prOnto:nominates” />
            <ruleml:Var keyref=”:edp” />
            <ruleml:Var keyref=”:y” />
            <ruleml:Var keyref=”:x” />
          </ruleml:Atom>
          <ruleml:Atom keyref=”:A121”>
            <ruleml:Rel iri=”prOnto:PersonalDataProcessing” />
            <ruleml:Var keyref=”:ep” />
            <ruleml:Var keyref=”:x” />
            <ruleml:Var keyref=”:z” />
          </ruleml:Atom>
          <ruleml:Atom>
            <ruleml:Rel iri=”prOnto:Purpose” />
            <ruleml:Var keyref=”:epu” />
          </ruleml:Atom>
          <ruleml:Atom>
            <ruleml:Rel iri=”prOnto:isBasedOn” />
            <ruleml:Var keyref=”:ep” />
            <ruleml:Var keyref=”:epu” />
          </ruleml:Atom>
          <ruleml:Atom>
            <ruleml:Rel iri=”prOnto:Consent” />
            <ruleml:Var keyref=”:c” />
          </ruleml:Atom>
          <ruleml:Atom keyref=”:A122”>
            <ruleml:Rel iri=”dapreco:GiveConsent” />
            <ruleml:Var keyref=”:ehc” />
            <ruleml:Var keyref=”:w” />
            <ruleml:Var keyref=”:c” />
          </ruleml:Atom>
          <ruleml:Atom keyref=”:A123”>
            <ruleml:Rel iri=”dapreco:AuthorizedBy” />
            <ruleml:Var keyref=”:eau” />
            <ruleml:Var keyref=”:epu” />
            <ruleml:Var keyref=”:c” />
          </ruleml:Atom>
        </ruleml:And>
      </ruleml:Exists>
    </ruleml:if>
    <ruleml:then>
      <ruleml:And>
        <ruleml:Atom>
          <ruleml:Rel iri=”prOnto:lawfulness” />
          <ruleml:Var keyref=”:ep” />
        </ruleml:Atom>
      </ruleml:And>
    </ruleml:then>
  </ruleml:Rule>
</lrml:ConstitutiveStatement>
```

As mentioned above, the connection between the predicates in the formulæ and the corresponding ontological concepts is implemented by the attribute `iri` of the tag `Rel`. In the DAPRECO knowledge base, there are three reference ontologies: `rioOnto`, `PrOnto`, and `dapreco`. `rioOnto` refers to the concepts of the logical framework, such as the concept of real existence at a certain time (predicate $RexistAtTime$) or the boolean connectives (*and*, *or*, and *not*). `PrOnto` refers to the concepts in the PrOnto ontology. Finally, `dapreco` includes the additional concepts which are not part of PrOnto but are needed to model the semantics of the GDPR norms; these predicates are connected to the ones in PrOnto via further constitutive rules.

It is then possible to navigate from a concept in the PrOnto ontology to the set of rules conveying the obligations and permissions related to that concept.

We developed a JavaScript tool to guide and monitor the building of the formulæ. Figure 3 shows the main screenshot of the tool.



Build complex Rio Formula

Rio Formula TAG preview:

<rioformula val="rioOnto:RexistAtTime(a1, t1), rioOnto:and(a1, ep, ec, eo, en), gdprOnto:dataController(y)"> 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,

Loaded predicates:

gdprOnto:dataProcessor(1)

Current predicate attribute template:

gdprOnto:dataProcessor(x)

Type new item:

Figure 3: The tool to edit reified Input/Output logic formulæ.

The tool allows to load an Akoma Ntoso file, as well as the set of predicates associated with the concepts of the referenced ontologies. Then, it allows to select an excerpt of text and to associate it with one or more reified Input/Output logic formulæ.

Since reified Input/Output logic formulæ are if-then rules of conjunctions of atomic predications, the human annotator simply specifies, one by one, the predicates in the conjunctions, while filling them with their arguments. Special placeholders allow to specify whether the rule is an obligation, a permission, or a constitutive rule, as well as which predicates belong to the LHS rather than to the RHS of the 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 ontological concepts. In future works, we plan to extend and reuse the tool for translating in reified Input/Output logic other legislative documents.

Of course, it is not possible to illustrate the details of each formula in the DAPRECO knowledge base. The following subsections illustrate how reification allows to easily and uniformly deal with three well-known thorny issues for the proper semantic-pragmatic representations of NL norms: nested obligations and nested permissions, defeasibility, and legal interpretations.

4.1 Nested obligations and nested permissions

As extensively argued above in subsection 3.1, reification allows to avoid nestings of subformulæ within complex operators in the ABox. Complex operators are represented in terms of first-order predicates that take, among their arguments, the main eventuality referring to the state of affairs described by the subformula. Further restrictions to model the operators's meaning may be asserted in terms of axioms in the TBox.

In light of this, concrete benefits brought by reification in the deontic realm can be shown by investigating deontic assertions featuring similar forms of nestings. Although such assertions are rare, they indeed exists, and they have been recently addressed, for instance, in (Governatori, 2015) (see section 5 below, devoted to related works). A preliminary example is provided in (17).

(17) 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 (17) 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, whenever this is the case. In this paper, we will term sentences such as (17) as "nested obligations"; in section 5 below, an example of nested permissions will be shown, i.e., sentence (48.e).

In standard Input/Output logic it is not possible to represent the meaning of nested obligations such as the one displayed above, as the formalism does not allow to specify semantic links between the if-then rules. Conversely, reification makes that possible, provided that we manage to explicitly assert an eventuality reifying *the fact that* someone is obliged to do something.

Suppose that, in our context, every manager who will attend the party "$\mathtt{p}$" is obliged to be elegant. Such an obligation may be represented via the formula in (18):

$$(18) \qquad \begin{array}{l}\forall_m \forall_t(\ \exists_{e_a}[(\mathit{RexistAtTime}\ e_a\ t) \wedge (\mathit{attend}'\ e_a\ m\ \mathtt{p}) \wedge (\mathit{manager}\ m)], \\ \qquad \exists_{e_e}[(\mathit{RexistAtTime}\ e_e\ t) \wedge (\mathit{elegant}'\ e_e\ m)]\ ) \in O\end{array}$$

We can then represent the status of being obliged, under which the manager m is, in terms of a new predication '$(\mathit{Obliged}'\ e_o\ t\ m\ e)$', meaning that m is obliged to the real existence of e at time t.

Then, we add to the knowledge base a special constitutive rule, associated with the rule in (18), asserting that *the fact that* the manager m is attending $\mathtt{p}$ entails that he or she is in the status of being obliged to be elegant:

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

According to this definition, the obligation of the manager's secretary in (17) is straightforwardly represent as:

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

In (20), '$\mathtt{secr}(m)$' is a FOL function that, taken a manager m as input, returns his secretary, who is the agent of the writing action e_w. Similarly, "$\mathtt{descr}(e_o)$" is a function that returns the description[28] of an eventuality; indeed, the secretary must write down a text describing the actions that the manager is obliged to do, not the actions themselves. On the other hand, the object/patient of the writing action is *the fact that* m is obliged to the real existence of e at time t.

Like *RexistAtTime*, *Obliged* is a possible *modality*[29] that eventualities may hold at a certain time t. *RexistAtTime* specifies which eventualities hold the status of real existence at t, while *Obliged* specifies which eventualities hold the status of obligatoriness at t, as well as who is the bearer[30] of such obligatory eventualities. A dual predicate *Permitted* may be of course introduced to specify which eventualities hold the status of permissiveness at t.

In the light of this, every obligation and every permission needs to be associated with an additional constitutive rule specifying the status of obligatoriness or permissiveness of their bearers. Note that this also applies recursively; for instance, (21) is the constitutive rule associated with (20), stating that *the fact that* the manager is obliged to the real existence of e entails that his secretary is in the status of being obliged to write this down in the manager's agenda:

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

As said at the beginning of this subsection, although nested obligations are rare, they could indeed occur in existing legislation[31]. The DAPRECO knowledge base contains some nested obligations as well. An example can be seen in Article 17, paragraph 2:

(22) `GDPR (Art.17, par.2)`: Where the controller has made the personal data public and is obliged pursuant to paragraph 1 to erase the personal data, the controller,

[28] We assume, for simplicity, that each eventuality may be described in a single way, i.e., that only a single description "$\mathtt{descr}(e)$" (functionally) corresponds to an eventuality e.

[29] See `http://www.isi.edu/~hobbs/bgt-modality.text`.

[30] LegalRuleML includes a special tag to mark the bearers of obligations and permissions: `<lrml:Bearer/>`. The DAPRECO knowledge base omits this tag to avoid redundancies: the bearers need to be already specified at the level of the underlying logical formalism, in order to enable nested obligations and nested permissions.

[31] (Idelberger et al., 2016) recently showed that nested obligations and nested permissions may also occur in the formalization of smart contracts.

> 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.

The reference provision in (22) is represented via the reified Input/Output logic formula[32] in (23):

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

The above formula reads as follows: if a controller y_1 controls public personal data z related to a data subject w, a controller y_2 controls a copy of z, and a situation really exists whereby y_1 is burdened by an obligation to make e_{ra1} really existing (where e_{ra1} is *the fact that* w has requested to erase his personal data z), then, in a certain moment t_2 in the future, y_1 is obliged to communicate e_{ra2} to y_2. In other words, y_1 is obliged to report to y_2 *the fact that* w has requested to also delete the copy of z. Furthermore, such a communication action has to be "reasonable"[33].

A final remark concerns that, although reification allows to flatten embeddings, the eventualities involved in the modal predicates define a hierarchy that parallels the same architecture of the embeddings. For instance, in (21) above, the eventuality e_{o2} "dominates" the eventuality e_{o1} through the modal predicate *Obliged′*, paralleling the fact that the obligation of the secretary "embeds" the obligation of the manager.

Of course, the logical framework must include axioms avoiding eventualities to dominate each other, to properly prevent the occurrence of self-reference paradoxes[34].

(Hobbs and Gordon, 2017) introduces such an axiom for the $RexistAtTime$ modality[35]. The axiom states that, for every predicate P, the following holds:

$$(24)\ \ \forall_{x_1 x_2 \ldots x_n}[\ (P\ x_1\ x_2\ \ldots\ x_n) \rightarrow \exists_e\exists_t[(\mathit{RexistAtTime}\ e\ t) \wedge (P'\ e\ \ x_1\ x_2\ \ldots\ x_n)]\ ]$$

The axiom in (24) relates the reification of the FOL predicates with their non-reified counterparts. As explained at the beginning of subsection 3.1, it states that, for instance, (25) entails (26):

[32] In reality, formula (23) is a simplification of the formula stored in the DAPRECO knowledge base and associated with (22), as it does not specify the *exceptions* of Article 17, paragraph 2. subsection 4.2 below illustrates how reified Input/Output logic deals with exceptions.

[33] As it will be clarified below in subsection 4.3, the truth value of the predicate '*reasonable*' depends on context-specific legal interpretations of the corresponding adjective.

[34] See `https://plato.stanford.edu/entries/self-reference`.

[35] See `https://www.isi.edu/~hobbs/bgt-evstruct.text`.

(25) $(\mathit{blond}\ \texttt{John})$

(26) $\exists_e \exists_t [(\mathit{RexistAtTime}\ e\ t) \wedge (\mathit{blond'}\ e\ \texttt{John})]$

Being *RexistAtTime* a FOL predicate itself, it is of course possible to recursively apply the axiom in (24), thus obtaining (27), (28), etc.

(27) $\exists_{e_1} \exists_e \exists_t [\ (\mathit{RexistAtTime}\ e_1\ t) \wedge (\mathit{RexistAtTime'}\ e_1\ e\ t) \wedge (\mathit{blond'}\ e\ \texttt{John})\ ]$

(28) $\exists_{e_2} \exists_{e_1} \exists_e \exists_t [\ (\mathit{RexistAtTime}\ e_2\ t) \wedge (\mathit{RexistAtTime'}\ e_2\ e_1\ t) \wedge$
$(\mathit{RexistAtTime'}\ e_1\ e\ t) \wedge (\mathit{blond'}\ e\ \texttt{John})\ ]$

Similar axioms must of course be asserted for the other modalities, e.g., *Obliged* and *Permitted*, but we omit their formalization.

Since the eventualities introduced via *RexistAtTime*, *Obliged*, *Permitted*, and so on will always be outscoped by the existential quantifiers, so that they will be different from the ones already present in the formula, a set of eventualities will never dominate each other in a cycle, i.e., no self-reference paradoxes occur.

4.2 Exceptions

Legal reasoning has to handle *conflicts*, in that often a rule derogates to another. The selection of applicable provisions in a certain case is a major problem in the general theory of law, and several criteria exist to determine which provisions prevail. The main ones are: chronological (subsequent laws derogate to the previous ones), hierarchical (primary provisions derogate to secondary ones), and specialty (special laws prevail over the general rule). So, for example, *general* regulative norms may be 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). To boot, exceptions may interact with each other, just like the general rules they are associated with. An example taken from Article 8, paragraph 1, of the GDPR is shown in (29):

(29) `GDPR (Art. 8, par. 1)`: Where point (a) of Article 6(1) 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.

`GDPR (Art. 5, par. 1)`: Personal data shall be:

(a) processed lawfully, fairly and in a transparent manner in relation to the data subject (“lawfulness, fairness and transparency”);

(b) . . .

> `GDPR (Art. 6, par. 1)`: 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;
>
> (b) ...

The intended meaning of (29) may be summarized as follows:

(30)

a. Processing of personal data ought to be lawful.

b. *The fact that* the data subject has given consent to the processing of his personal data entails that the processing is lawful.

c. Exception to point (b): in case the data subject is less than 16 years old, his consent does not entail lawfulness of processing.

d. In case the data subject is less than 16 years old, *the fact that* the holder of his parental responsibility has given consent to the processing of his personal data, entails that the processing is lawful.

e. Exception to points (c) and (d): Member States may lower the minimal age for giving consent, although not below 13 years.

(30.a–b) concern the representation of Article 6, paragraph 1, point (a). The corresponding formulæ in reified Input/Output logic, which do not seem to require further explanations, are shown in (31) and (32), respectively. Note that the former is an obligation, while the latter is a constitutive rule, used to define the lawfulness of processing.

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

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

A standard solution to represent the exception in (30.c) is to introduce additional constructs that implement *defeasibility.* Several approaches in Legal Informatics, such as (Parent, 2011), use priorities and/or partial orders to *explicitly* assert when a norm is "stronger" than another one, that is, when it should prevail.

In (Hobbs and Gordon, 2017), defeasibility is handled *implicitly* by means of constructs drawn from Circumscriptive Logic (McCarthy, 1980). We illustrate the simple idea behind Circumscriptive Logic with an example[36]. The fact that every bird flies may be represented in FOL as '$\forall_x[bird(x) \rightarrow fly(x)]$'. We add another predicate '$normalBF$' to the LHS to render the formula defeasible. This predicate models the fact that birds fly only when it is "normal" to assume so. The resulting expression is '$\forall_x[(bird(x) \wedge normalBF(x)) \rightarrow fly(x)]$'.

To model that penguins are non-flying birds, we assert they are not normal with respect the property of flying: '$\forall_x[\,penguin(x) \rightarrow \neg normalBF(x)]$'. Based on this assertion, we may consistently state that they do not fly: '$\forall_x[\,penguin(x) \rightarrow (bird(x) \wedge \neg fly(x))]$'.

In other words, predicates like '$normalBF$' are introduced in the formulæ in order to "block" the general inferences, and to consistently assert more specific ones. In this sense, the specific rules are stronger than the general ones, meaning that they have "higher priority".

A similar mechanism handles exceptions. General inferences are allowed only if the exceptions given by the specific rules do *not* occur. Exceptions correspond to FOL predications paralleling '$normalBF(x)$' in the previous example. However, to avoid explicitly asserting whether an exception does not hold, instead of using a simple negation, it is possible to use a negation-as-failure (naf, for short), which is a construct supported by LegalRuleML. The distinction between '$\neg A$' and '$naf(A)$' can be described as follows: the former is true when it can be derived that A is false, while the latter is true when it cannot be derived that A is true. In other words, the latter is true both when A is false *and* when A is unknown.

In light of this, the formula in (32) is rewritten as in (33). 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[37].

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

[36] Taken from `https://www.isi.edu/~hobbs/bgt-defeasibility.text`.

[37] In footnote 32 above, we noted that the two elements x and y of a pair $(x,\ y)$ belonging to either O or P are always made up of conjunctions of atomic predications, *with an important exception.* The exception is represented by 'naf', which indeed introduces one level of nesting in the formulæ.

Using LegalRuleML, '$naf(exceptionCha2Art8Par1\, e_p)$' is codified as follows:

```
<ruleml:Naf>
  <ruleml:Atom>
    <ruleml:Rel iri="rioOnto:exceptionCha2Art8Par1" />
    <ruleml:Var keyref=":ep" />
  </ruleml:Atom>
</ruleml:Naf>
```

The exception in (30.c) can then be modeled by using the formula in (34), which entails the exception '$(exceptionCha2Art8Par1\, e_p)$', in case the data subject is less than 16 years old, and "blocks" the inference in (33). New obligations may be then consistently asserted for data subjects who are less than 16 years old. An example is the one corresponding to (30.d), but it will be omitted from this description.

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

Finally, it is necessary to (recursively) define an exception in (34) as well, since Article 8, paragraph 1, allows Member States to lower down the minimal age for giving consent (although not below 13 years).

To avoid over-assertion of rules, we can substitute the constant '16' in (34) with a FOL function '`minAgeForConsent`' that takes in input the processing action e_p and returns the minimal age for giving consent to e_p in the context where the formula is instantiated. In other words, we replace the last predication in the LHS of formula (34) with ('$\mathit{numeric\text{-}less\text{-}than}\ \ \texttt{ageOf}(w)\ \ \texttt{minAgeForConsent}(e_p)$)'.

The following constitutive rule defines that the value of minimal age for giving consent to processing of personal data is 16, provided that there are no exceptions.

$$\begin{aligned}
(35)\ \forall_{e_p}(\ & naf(\mathit{exceptionMinAgeForConsent}\ e_p), \\
& (\mathit{numeric\text{-}equal}\ \ \texttt{minAgeForConsent}(e_p)\ \ 16)\ )\in C
\end{aligned}$$

Assuming, for instance, that French national legislation lowers the minimal age for giving consent to the processing of personal data down to 14 years, the DAPRECO knowledge base can then be enriched with the following constitutive rule:

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

4.3 Legal interpretations

Handling multiple interpretations of legal provisions is perhaps the best known problem in Legal Informatics. Laws can be pragmatically interpreted in multiple, and often incompatible, ways in different contexts. Furthermore, what makes legal texts so much dependent on human interpretation is that they are used in disputes that represent different interests, so that the interpretation of the provisions tends to be stretched depending on the interest involved.

Since it is impossible to predict every possible context where the provisions will be deployed, legislators tend to use terms that are flexible enough to be adapted to the required socio-legal context, and sometimes may appear as *vague*, such as “reasonable” in the example shown in (22). In other words, such terms are used in case the legislator cannot account for the multitude of situations that should be covered by the abstract legislation, which often depend on the legal cases as they occur, and on the reflections of legal doctrine. It is eventually up to judges and other appointed authorities to decide the “final” interpretation of provisions in each context. However, even in similar contexts, it is quite common that different judges adopt different legal interpretations, incompatible among themselves (sometimes even concerning identical cases).

LegalRuleML provides tags, illustrated below, to enable the assertion of multiple legal interpretations, while specifying that they are mutually exclusive of each another.

Within the DAPRECO project we provide some possible, early legal interpretations of GDPR provisions in terms of *correlations* between them and the controls in some ISO security standards (see (Bartolini et al., 2016)).

A clear example is shown in (37). The DAPRECO knowledge base assumes that provisions (37.a-b) are correlated, so that compliance with control A9.1 of the ISO/IEC 27018:2014 security standard in (37.b) entails compliance with Article 33, paragraph 2, of the GDPR, in (37.a-b).

(37)

a. `GDPR (Art.33, par.2)`: The processor shall notify the controller without undue delay after becoming aware of a personal data breach.

b. `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.

In reified Input/Output logic, (37.a-b) are formalized as in (38) and (39) respectively[38].

[38] In (39), we formalized “the relevant cloud service customer” occurring in (37) via the predicate *PIIController*. According to ISO 27018, Article 0.1: “The cloud service customer, who has the contractual relationship with the public cloud PII processor, can range from a natural person, a ‘PII principal’, processing his or her own PII in the cloud, to an organization, a ‘PII controller’, processing PII relating to many PII principals”.

$$\begin{aligned}
&(38)\ \forall_x \forall_y \forall_{e_b} \forall_{t_1} (\\
&\quad \exists_{a_1} \exists_{e_{dp}} \exists_{e_p} \exists_{e_a} \exists_w \exists_z [(\mathit{RexistAtTime}\ a_1\ t_1) \wedge (\mathit{and}\ a_1\ e_{dp}\ e_p\ e_a) \wedge\\
&\qquad\quad (\mathit{DataSubject}\ w) \wedge (\mathit{PersonalData}\ z\ w) \wedge (\mathit{Controller}\ y\ z) \wedge\\
&\qquad\quad (\mathit{Processor}\ x) \wedge (\mathit{nominates}\text{'}\ e_{dp}\ y\ x) \wedge\\
&\qquad\quad (\mathit{PersonalDataProcessing}\text{'}\ e_p\ x\ z) \wedge\\
&\qquad\quad (\mathit{AwareOf}\text{'}\ e_a\ x\ e_b) \wedge (\mathit{DataBreach}\ e_b\ z)],\\
&\quad \exists_{e_n} \exists_{t_2} [(\mathit{RexistAtTime}\ e_n\ t_2) \wedge (\mathit{numeric\text{-}greater\text{-}than\text{-}or\text{-}equal}\ t_2\ t_1) \wedge\\
&\qquad\quad (\mathit{Communicate}\text{'}\ e_n\ x\ y\ \mathtt{allInfoAbout}(e_b)) \wedge (\mathit{nonDelayed}\ e_n)]\ ) \in O
\end{aligned}$$

$$\begin{aligned}
&(39)\ \forall_x \forall_y \forall_{e_a} \forall_{t_1} (\\
&\quad \exists_{a_1} \exists_{e_{dp}} \exists_{e_p} \exists_w \exists_z \exists_k [(\mathit{RexistAtTime}\ a_1\ t_1) \wedge (\mathit{and}\ a_1\ e_{dp}\ e_p\ e_a) \wedge\\
&\qquad (\mathit{PIIPrincipal}\ w) \wedge (\mathit{PIIController}\ y\ z) \wedge (\mathit{PIIProcessor}\ x) \wedge\\
&\qquad (\mathit{PII}\ z\ w) \wedge (\mathit{nominates}\text{'}\ e_{dp}\ y\ x) \wedge (\mathit{PersonalDataProcessing}\text{'}\ e_p\ x\ z) \wedge\\
&\qquad (\mathit{access}\text{'}\ e_a\ k\ z) \wedge (\mathit{unauthorized}\ e_a)],\\
&\quad \exists_{e_n} \exists_{t_2} [(\mathit{RexistAtTime}\ e_n\ t_2) \wedge (\mathit{numeric\text{-}greater\text{-}than\text{-}or\text{-}equal}\ t_2\ t_1) \wedge\\
&\qquad (\mathit{Communicate}\text{'}\ e_n\ x\ y\ \mathtt{allInfoAbout}(e_a)) \wedge (\mathit{promptly}\ e_n)]\ ) \in O
\end{aligned}$$

Correlating (39) and (38) amounts to introducing new assertions into the knowledge base, containing further entailments between the predicates occurring in the two formulæ, so that one obligation is satisfied (or violated) if the other one is.

It seems rather unquestionable to assume that “PII” (Personally Identifiable Information), a term regularly used in ISO/IEC 27018:2014, and “personal data”, which is one of the core terms of the GDPR, refer to the same concept. The definitions provided for the two terms are also essentially identical. Thus, the knowledge base may safely include the implication ‘$\forall_z \forall_w [(\mathit{PII}\ z\ w) \rightarrow (\mathit{PersonalData}\ z\ w)]$’, in the form of a constitutive rule, in order to assert that PII, as defined in ISO/IEC, has been considered as personal data, as defined in the GDPR.

Similarly, we may add to the knowledge base, without particular concern, the formulæ ‘$\forall_w [(\mathit{PIIPrincipal}\ w) \rightarrow (\mathit{DataSubject}\ w)]$’, ‘$\forall_x [(\mathit{PIIProcessor}\ x) \rightarrow (\mathit{Processor}\ x)]$’, ‘$\forall_y \forall_z [(\mathit{PIIController}\ y\ z) \rightarrow (\mathit{Controller}\ y\ z)]$’, and ‘$\forall_e [(\mathit{promptly}\ e) \rightarrow (\mathit{nonDelayed}\ e)]$’, 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”.

On the other hand, it could be questionable to assume that an unauthorized access *count as* a data breach. In other words, the constitutive rule in (40) may be subject to different legal interpretations.

$$\begin{aligned}
&(40)\ \forall_{e_a} \forall_z \forall_t (\ \exists_k [(\mathit{RexistAtTime}\ e_a\ t) \wedge (\mathit{access}\text{'}\ e_a\ k\ z) \wedge (\mathit{unauthorized}\ e_a)],\\
&\qquad\qquad (\mathit{DataBreach}\ e_a\ z)\ ) \in C
\end{aligned}$$

To model different legal interpretations of (40), we use the same mechanism shown previously to model exceptions: we add a special predication '$(assumption\, e_a)$', which is true if it may be *assumed* that the inference in (40) is valid. The final version of the constitutive rule is therefore the following:

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

The knowledge base then contains further constitutive rules that *block* the inference in (40), i.e., that entail the *negation* of $(assumption\, e_a)$.

It is also possible to introduce constitutive rules that determine when $(assumption\, e_a)$ is *true*, even if these rules will be redundant: $(assumption\, e_a)$ is already assumed to be *always* true, unless it is explicitly written that it is false. The reason behind this, as argued in (Robaldo and Sun, 2017), is that reified Input/Output logic has been designed for building knowledge bases "able to keep track of the different legal interpretations over time". Therefore, if some legal authorities explicitly state that the default assumptions are true, we should "register" this in terms of parallel explicit formulæ in the knowledge base, even if these formulæ are redundant. Otherwise, we lose the information that these legal authorities "confirmed" the assumptions.

As an example, we can assume three fictitious pieces of case law, not pertaining to actual legal decisions, for the sole purpose of illustrating how legal interpretations are codified within the DAPRECO knowledge base. Suppose we have the following legal interpretations of (41):

(42)

a. *Italian Corte di Cassazione, sezione civile, 12530/2012*: An unauthorized access *counts as* a data breach, according to the definitions found in the state of the art of the cybersecurity and data protection domains.

b. *Spanish Audiencia provincial de Toledo, n. 57/2016, 2/12/2016*: An unauthorized access does *not* count as 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.

c. *French Tribunal de Grande Instance d'Avignon, decision du 17/04/2016*: In this case, we assume that the specific conditions examined by the Tribunal consisted in the company "Alpha" performing a security test on an IT system; even if unauthorized accesses indeed took place, and although they have to be considered as data breaches in the general case, in this specific scenario they cannot be taken as such in that they were part of the security test.

The three legal interpretations above are modeled using the three sets of reified Input/Output logic in (43), (44), and (45) respectively.

(43) $$\forall_{e_a}(\ \exists_k\exists_z\exists_t[(RexistAtTime\ e_a\ t) \wedge (access\text{'}\ e_a\ k\ z) \wedge (unauthorized\ e_a)],\ (assumption\ e_a)\ ) \in C$$

$$(44)\ \forall_{e_a}(\ \exists_k\exists_z\exists_t[(\mathit{RexistAtTime}\ e_a\ t) \wedge (\mathit{access}\text{'}\ e_a\ k\ z) \wedge (\mathit{unauthorized}\ e_a)],$$

$$\neg(\mathit{assumption}\ e_a)\ ) \in C$$

$$(45)\ \forall_{e_a}(\ \exists_k\exists_z\exists_t[(\mathit{RexistAtTime}\ e_a\ t) \wedge (\mathit{access}\text{'}\ e_a\ k\ z) \wedge (\mathit{unauthorized}\ e_a) \wedge$$
$$\mathit{naf}(\mathit{exceptionSecurityTest}\ e_a)],$$
$$(\mathit{assumption}\ e_a)\ ) \in C$$

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

$$(\mathit{exceptionSecurityTest}\ e_a)\ ) \in C$$

The formulæ in (43) and (44) have the same LHS of (40), and they respectively entail '($\mathit{assumption}\ e_a$)' and its negation.

As discussed above, (43) is indeed redundant, in that we already postulated that ($\mathit{assumption}\ e_a$) is always true, provided that it is not explicitly asserted as false; however, we want to explicitly encode that the Italian court supports the assumption.

On the other hand, (44) is a simplification in that it only codifies that the Spanish court does not accept ($\mathit{assumption}\ e_a$), while it does not codify the full legal interpretation of data breach by that court, i.e., that a data breach includes both an unauthorized access, a breach of security, and a causal connection between them; however, such details are not relevant for the present discussion.

Finally, (45) includes *two* constitutive rules, one of which entails an exception to the other one. The two formulæ in (45) read as follows: although the court recognizes that an unauthorized access counts as a data breach in the general case, when it results from security tests, such as the ones performed by the company "Alpha", it is not to be considered as such. Security tests are then exceptions to the general case.

The crucial difference between (43) and (45) is that, according to the former, the processor shall comply with its GDPR obligations, including the one of notifying the controller without undue delay, also in the case of security tests. In such cases, the notification will simply specify that the unauthorized accesses were intentional and under the control of the processor itself. On the other hand, according to (45), processors are exempt from their GDPR obligations in case of security tests.

It is clear that the three legal interpretations in (42) cannot hold at the same time, as their conjunction is inconsistent. LegalRuleML provides the tag '`<lrml:Alternatives>`' to explicitly assert that they are mutually exclusive (Athan et al., 2014). We then codify each of the three formulæ above into a tag `<lrml:ConstitutiveStatement>`, similarly to what was done above in Listing 1, assigning each a different `key`:

```
<lrml:ConstitutiveStatement key="CassazioneCivileItalia">
  ...
</lrml:ConstitutiveStatement>
<lrml:ConstitutiveStatement key="AudienciaDeToledoSpain">
  ...
</lrml:ConstitutiveStatement>
<lrml:ConstitutiveStatement key="TribunalDAvignonFrance">
  ...
</lrml:ConstitutiveStatement>
```

The `keys` are then referred to within the LegalRuleML tag `<lrml:Alternatives>`, which specifies that each of the three constitutive rules is an “alternative” to the other ones, meaning that they are all mutually exclusive among themselves.

```
<lrml:Alternatives>
  <lrml:hasAlternative keyref=”#CassazioneCivileItalia”/>
  <lrml:hasAlternative keyref=”#AudienciaDeToledoSpain”/>
  <lrml:hasAlternative keyref=”#TribunalDAvignonFrance”/>
</lrml:Alternatives>
```

5 Related works

Reified Input/Output logic (Robaldo and Sun, 2017) has been proposed as a logical framework to formalize provisions in existing legislation, which are expressed in natural language only. Reified Input/Output logic takes the neo-Davidsonian approach of (Hobbs and Gordon, 2017) as the object logic of two sequential Input/Output systems, described above in Figure 2, one for constitutive rules and one for regulative rules.

(Hobbs and Gordon, 2017) features a total avoidance of nestings, thus allowing to manage complex phenomena in Natural Language Semantics in a simple and flexible way, and so does reified Input/Output logic. This holds for Natural Language Semantics in general, not only for legal texts. It has been argued in (Hobbs, 1998) and (Hobbs, 2001) that avoidance of nestings allows for a straightforward treatment of anaphora, while (Robaldo, 2010a) and (Robaldo, 2010b) argue that embeddings should be *always* avoided, in order to properly represent not only the predications, but also the sets of individuals involved in the predications.

Therefore, as explained in the two subsections below, the crucial benefits brought by reification to Input/Ouput logic are formal simplicity and modularity, achieved without lowering the required expressivity. Formal simplicity and modularity allow for the easy and quick building of large repositories of reified Input/Output formulæ, such as the DAPRECO knowledge base.

5.1 Exceptions and legal interpretations

Exceptions and legal interpretations have been well-known problems in Legal Informatics for decades (see (MacCormick and Summers, 1991; Nute, 1997)). Literature is huge, and several formalizations (Boella et al., 2010; Antoniou et al., 2001; Prakken, 2005; Sartor, 2005; Governatori et al., 2009; Brozek, 2014; Walton, Sartor, and Macagno, 2016; Malerba, 2017) have been proposed to deal with them.

All mentioned approaches, as well as the one proposed in this paper, deal with exceptions and legal interpretations via some kind of *defeasible* logic. Mechanisms imported from standard Default Logic (Reiter, 1987), priorities, or partial orders between normative rules are common ways to implement defeasibility.

(Parent, 2011) proposes a defeasible extension of standard Input/Output logic based on priorities in order to properly handle moral conflicts.

On the other hand, in reified Input/Output logic, as in (Hobbs and Gordon, 2017), defeasibility is implemented in terms of mechanisms imported from (McCarthy, 1980) that allow to “block” the entailments associated with the constitutive rules.

As explained in subsection 6.2 below, devoted to future works, the latter are technically equivalent to standard constructs proposed since the Nineties to undercut defeasible rules in logic programming (Nute, 1994a), and still used nowadays (Governatori et al., 2013; Amgoud and Nouioua, 2015). The subsection will address the need of an exhaustive analysis of the expressivity and the computational complexity of defeasible Input/Output logic frameworks, such as (Parent, 2011) and the one proposed here.

Exceptions and legal interpretations are indeed two sides of the same coin, related to the fact that existing provisions are highly dependent on human subjectivity and that all possible real contexts where they could be applied cannot be predicted *a priori*.

Exceptions are mostly related to the notion of *preference* between multiple valid rules, using the prevalence criteria that have been briefly mentioned at the beginning of subsection 4.2. For example, considering the specialty criterion, special rules have to be preferred over general ones; however, in case their LHS does not hold, the general rule applies.

On the other hand, legal interpretation of the provisions mostly concerns the occurrence of multiple rules that cannot be applied together, because the conjunctions of their RHSs is inconsistent, so that they are *mutually exclusive*.

It is of course possible to find real-world scenarios showing preferences among multiple valid interpretative rules, which are exceptions to general legal interpretations. Such a scenario has been exemplified above in (45). According to the (fictional) case law issued at the Audiencia provincial de Toledo, unauthorized accesses are interpreted as data breaches (general rule), unless they are specifically due to security tests (special rule).

Formula (45) shows that the reified Input/Output logic mechanism to implement defeasibility also allows to easily represent preferences among legal interpretations.

Similarly, (Rotolo, Governatori, and Sartor, 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 not used to derive new conclusions, but only to "block" others; defeaters are similar to our mechanism, imported from Circumscriptive Logic, exemplified for instance above in (34). 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 Input/Output logic, it is not necessary to define two different formal schemas to distinguish legal interpretations at either the sentential or the intra-sentential level, in that reification allows to uniformly move across different levels of abstraction.

To understand why, let us consider the reified Input/Output logic formula in (23) again, corresponding to the GDPR provision in (37.a), copied again in (46) for the reader's convenience:

(46) `GDPR (Art.33, par.2)`: The processor shall notify the controller without undue delay after becoming aware of a personal data breach.

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

subsection 4.3 already introduced an example related to the (fictitious) legal interpretation of the term “data breach” in contexts where unauthorized accesses occur. For some legal authorities, unauthorized accesses are considered as data breaches, for others they are not.

On the other hand, in order to ascribe different legal interpretations at the sentential level, we need to assert additional constitutive rules that defeasibly entail the truth or the falsity of the whole LHS of the formula above. For instance, let’s assume a fictional context where the processor x has been informed about a data breach by a friend k via oral communication. Some legal authorities may assume that oral communication is sufficient to entail that the processor has been properly made aware of the data breach, and consequently that the LHS of (46) holds in such contexts, while other authorities may decide otherwise.

To model these different legal interpretations, the constitutive rule in (47) is added to the knowledge base. Additional constitutive rules may be added as well, to distinguish the legal authorities for which ‘$(assumption\ e_{cm})$’ is true from the ones for which it is not, along the line explained above in subsection 4.3.

(47)
$$\begin{aligned}
&\forall_x \forall_{e_b} \forall_{t_1} (\\
&\quad \exists_{a_1} \exists_{e_{dp}} \exists_{e_p} \exists_{e_{cm}} \exists_y \exists_z \exists_w \exists_k [(RexistAtTime\ a_1\ t_1) \wedge (and\ a_1\ e_{dp}\ e_p\ e_{cm}\ e_b) \wedge\\
&\qquad\quad (DataSubject\ w) \wedge (PersonalData\ z\ w) \wedge (Controller\ y\ z) \wedge\\
&\qquad\quad (Processor\ x) \wedge (nominates'\ e_{dp}\ y\ x) \wedge\\
&\qquad\quad (PersonalDataProcessing'\ e_p\ x\ z) \wedge (DataBreach\ e_b\ z) \wedge\\
&\qquad\quad (Communicate'\ e_{cm}\ k\ x\ e_b) \wedge (oralForm\ e_{cm}) \wedge (assumption\ e_{cm})],\\
&\quad \exists_{e_a} [(RexistAtTime\ e_a\ t_1) \wedge (AwareOf'\ e_a\ x\ e_b)]\ ) \in C
\end{aligned}$$

Contrary to (Governatori and Rotolo, 2008), reified Input/Output logic is then able to both add assumptions (and the corresponding constitutive rules) at the sentential level, *and* other assumptions (and the corresponding constitutive rules) at the chunk/word level, using a single uniform mechanism.

5.2 Nested obligations and nested permissions

Readings concerning nested obligations and nested permissions (see subsection 4.1 above) 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 and nested 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 and nested permissions through Linear Temporal Logic inferences. Specifically, the analysis focuses on the Australian Privacy Amendment (Enhancing Privacy Protection) Bill 2012[39], which contains provisions that have the same logical structure of sentences (48.a-e).

(48)

a. Collection of information of type A is forbidden, in the general case.

b. Collection of information of type A is permitted if there is a court order authorizing it. This means that collections of information of type A under court orders are exceptions to the previous rule.

c. Destruction of illegaly collected information of type A, before accessing it, compensates the prohibition in (48.a).

d. Collection of information of type B is forbidden, in the general case.

e. Collection of information of type B is permitted whenever collection of information of type A is permitted. This means that permissions of collecting information of type A provide exceptions to the previous rule.

In (48.a-e), information of type A and information of type B are *disjoint*, i.e., $A \cap B \equiv \emptyset$.

Note that (48.e) contains a nested permission. On the other hand, (48.c) is a so-called "compensatory clause": destruction of information of type A, unlawfully collected, before accessing it *compensates* the violation of (48.a). In other words, in case someone collects information of type A but then destroys it prior to accessing it, (48.a) is not indeed violated, and the behaviour is compliant with the norms.

(Governatori, 2015) shows that, under Linear Temporal Logic derivations, the scenario where one collects both information of type A *and* information of type B, and he or she later destroys the unlawfully collected information of type A, is compliant with the provisions in (48.a-e). However, this is incorrect, as the rule in (48.d) was violated: there was no permission to collect information of type B, and compensation of (48.a) does not entail compensation of (48.d). Therefore, in this scenario, Linear Temporal Logic is indeed unable to derive the violation of (48.d).

[39] `https://www.legislation.gov.au/Details/C2012B00077/ExplanatoryMemorandum/Text`.

This paper does not address compensations, as they do not occur in the DAPRECO knowledge base. A simple and intuitive way to model violations and compensations in reified Input/Output logic, which seems to be acceptable at least for the general cases such as the exemplified one (see (Boella and van der Torre, 2004a) for a general discussion), amounts at defining two new predicates, '$Violated$' and '$Compensated$', that respectively model the reification of *the fact that* an obligation has been violated and *the fact that* an obligation has been compensated. The following (universal) constitutive rule axiomatizes the relation between the two predicates:

$$\begin{aligned}
(49)\quad & \forall_{e_o}\forall_x\forall_t(\ \exists_e\exists_{e_n}[(\mathit{Obliged}'\ e_o\ t\ x\ e) \wedge (\mathit{RexistAtTime}\ e_n\ t) \wedge (\mathit{not}\ e_n\ e) \wedge \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad naf(\mathit{Compensated}\ e_o\ t)], \\
& \qquad \exists_{e_v}[(\mathit{Violated}\ e_v\ t\ x\ e_o)]\ ) \in C
\end{aligned}$$

The rule in (49) reads as follows: if x is obliged to make e really existing at time t, but, on the contrary, at time t the negation of e really exists and it may be assumed that x's obligation has not been compensated, then x's obligation is violated at time t.

Having defined the concepts of violation and compensation of an obligation, (48.a-e) can be respectively represented via the reified Input/Output logic formulæ from (50) to (54). Note that the predicate *Compensated* occurs in formula (52).

$$\begin{aligned}
(50)\quad & \forall_{e_{co}}\forall_t(\ \exists_x\exists_y[(\mathit{infoA}\ y) \wedge naf(\mathit{exceptionCourtOrder}\ e_{co}\ x\ t) \wedge (\mathit{Collect}'\ e_{co}\ x\ y)], \\
& \qquad \exists_{e_{nc}}[(\mathit{RexistAtTime}\ e_{nc}\ t) \wedge (\mathit{not}\ e_{nc}\ e_{co})]\ ) \in O
\end{aligned}$$

$$\begin{aligned}
& \forall_{e_{co}}\forall_t(\ \exists_x\exists_y[(\mathit{infoA}\ y) \wedge naf(\mathit{exceptionCourtOrder}\ e_{co}\ x\ t) \wedge (\mathit{Collect}'\ e_{co}\ x\ y)], \\
& \qquad \exists_{e_o}\exists_{e_{nc}}[(\mathit{Obliged}'\ e_o\ t\ x\ e_{nc}) \wedge (\mathit{not}\ e_{nc}\ e_{co})]\ ) \in C
\end{aligned}$$

$$\begin{aligned}
(51)\quad & \forall_x\forall_{e_{co}}\forall_t(\ \exists_{e_{au}}\exists_w\exists_y[(\mathit{RexistAtTime}\ e_{au}\ t) \wedge (\mathit{Authorize}'\ e_{au}\ w\ e_{co}) \wedge \\
& \qquad\qquad\qquad\qquad\qquad (\mathit{courtOrder}\ w) \wedge (\mathit{Collect}'\ e_{co}\ x\ y) \wedge (\mathit{infoA}\ y)], \\
& \qquad (\mathit{exceptionCourtOrder}\ e_{co}\ x\ t)\ ) \in C
\end{aligned}$$

$$\begin{aligned}
& \forall_{e_{co}}\forall_t(\ \exists_{e_{au}}\exists_x\exists_y\exists_w[(\mathit{RexistAtTime}\ e_{au}\ t) \wedge (\mathit{Authorize}'\ e_{au}\ w\ e_{co}) \wedge \\
& \qquad\qquad\qquad\qquad\qquad (\mathit{courtOrder}\ w) \wedge (\mathit{Collect}'\ e_{co}\ x\ y) \wedge (\mathit{infoA}\ y)], \\
& \qquad (\mathit{RexistAtTime}\ e_{co}\ t)\ ) \in P
\end{aligned}$$

$$\begin{aligned}
& \forall_{e_{co}}\forall_x\forall_t(\ \exists_{e_{au}}\exists_y\exists_w[(\mathit{RexistAtTime}\ e_{au}\ t) \wedge (\mathit{Authorize}'\ e_{au}\ w\ e_{co}) \wedge \\
& \qquad\qquad\qquad\qquad\qquad (\mathit{courtOrder}\ w) \wedge (\mathit{Collect}'\ e_{co}\ x\ y) \wedge (\mathit{infoA}\ y)], \\
& \qquad \exists_{e_p}[(\mathit{Permitted}'\ e_p\ t\ x\ e_{co})]\ ) \in C
\end{aligned}$$

$$\begin{aligned}
(52)\quad & \forall_{e_o}\forall_{t_1}(\ \exists_x\exists_y\exists_{e_{nc}}\exists_{e_{co}}\exists_{e_{de}}\exists_{t_2}[\ (\mathit{Obliged}'\ e_o\ t_1\ x\ e_{nc}) \wedge (\mathit{not}\ e_{nc}\ e_{co}) \wedge \\
& \qquad\qquad\qquad (\mathit{Collect}'\ e_{co}\ x\ y) \wedge (\mathit{infoA}\ y) \wedge (\mathit{RexistAtTime}\ e_{de}\ t_2) \wedge \\
& \qquad\qquad\qquad (\mathit{numeric\text{-}greater\text{-}than\text{-}or\text{-}equal}\ t_2\ t_1) \wedge (\mathit{Destroy}'\ e_{de}\ x\ y)\ ], \\
& \qquad (\mathit{Compensated}\ e_o\ t_1)\ ) \in C
\end{aligned}$$

(53) $\forall_{e_{co}}\forall_{t}(\ \exists_{x}\exists_{y}[(infoB\ y) \wedge (Collect\text{'}\ e_{co}\ x\ y)],$
$\qquad\exists_{e_{nc}}[(RexistAtTime\ e_{nc}\ t) \wedge (not\ e_{nc}\ e_{co})]\ ) \in O$

$\forall_{e_{co}}\forall_{t}(\ \exists_{x}\exists_{y}[(infoB\ y) \wedge (Collect\text{'}\ e_{co}\ x\ y)],$
$\qquad\exists_{e_{o}}\exists_{e_{nc}}[(Obliged'\ e_{o}\ t\ x\ e_{nc}) \wedge (not\ e_{nc}\ e_{co})]\ ) \in C$

(54) $\forall_{x}\forall_{t}(\ \exists_{e_p}\exists_{e_{co1}}\exists_{y}[(Permitted'\ e_{p}\ t\ x\ e_{co1}) \wedge (Collect\text{'}\ e_{co1}\ x\ y) \wedge (infoA\ y)],$
$\qquad\exists_{e_{co2}}\exists_{z}[(RexistAtTime\ e_{co2}\ t) \wedge (Collect\text{'}\ e_{co2}\ x\ z) \wedge (infoB\ z)\ ) \in P$

$\forall_{x}\forall_{t}(\ \exists_{e_{p1}}\exists_{e_{co1}}\exists_{y}[(Permitted'\ e_{p1}\ t\ x\ e_{co1}) \wedge (Collect\text{'}\ e_{co1}\ x\ y) \wedge (infoA\ y)],$
$\qquad\exists_{e_{p2}}\exists_{e_{co2}}\exists_{z}[(Permitted'\ e_{p2}\ t\ x\ e_{co2}) \wedge (Collect\text{'}\ e_{co2}\ x\ z) \wedge$
$\qquad\qquad(infoB\ z)]\ ) \in C$

The problematic scenario exemplified above is described by the following atoms (let `John` be the subject collecting the data)[40]:

(55) $(RexistAtTime\ \mathtt{e_{caj}}\ \mathtt{t_1}) \wedge (Collect\text{'}\ \mathtt{e_{caj}}\ \mathtt{John}\ \mathtt{y}) \wedge (infoA\ \mathtt{y}) \wedge$
$(RexistAtTime\ \mathtt{e_{cbj}}\ \mathtt{t_1}) \wedge (Collect\text{'}\ \mathtt{e_{cbj}}\ \mathtt{John}\ \mathtt{z}) \wedge (infoB\ \mathtt{z}) \wedge$
$(RexistAtTime\ \mathtt{e_{daj}}\ \mathtt{t_2}) \wedge (Destroy\text{'}\ \mathtt{e_{daj}}\ \mathtt{John}\ \mathtt{y}) \wedge (infoA\ \mathtt{y}) \wedge$
$(numeric\text{-}greater\text{-}than\text{-}or\text{-}equal\ \mathtt{t_2}\ \mathtt{t_1})$

It is easy to see that (55) is not compliant with the if-then rules from (49) to (54). From the constitutive rule in (50) and the input in (55), we derive:

$$\exists_{e_o}\exists_{e_{nc}}[(Obliged'\ e_o\ \mathtt{t_1}\ \mathtt{John}\ e_{nc}) \wedge (not\ e_{nc}\ \mathtt{e_{caj}})]$$

Stating that, at time $\mathtt{t_1}$, John is obliged to not collect the portion of information of type A referred by '`y`'. Note that, in (50), '$naf(exceptionCourtOrder\ \mathtt{e_{caj}}\ \mathtt{John}\ \mathtt{t_1})$' evaluates to true, in that we do not have any knowledge about a court order authorizing the collecting action '$\mathtt{e_{caj}}$'.

On the other hand, the inference in (49) is blocked precisely because the literal '$naf(Compensated\ e_o\ \mathtt{t_1})$' evaluates to false, for the obligation e_o. In other words, '$(Compensated\ e_o\ \mathtt{t_1})$' is derived through (52), in that there is a destroying action of the illegally collected information of type A, performed by John at time $\mathtt{t_2} >= \mathtt{t_1}$.

Therefore, a violation for the illegally collected information of type A is not derived. However, we can still derive the violation of the illegally collected information of type B, through (49), (53), and (55).

Thus, like the approach of (Governatori and Rotolo, 2006), reified Input/Output logic also derives the proper inferences within the exemplified scenario. Thanks to reification, the proper violations and compensations are naturally and intuitively derived through basic FOL implications.

[40] Note that, in (55), $\mathtt{e_{caj}}$, $\mathtt{e_{cbj}}$, $\mathtt{e_{daj}}$, $\mathtt{t_1}$, $\mathtt{t_2}$, `John`, `y`, and `z` are FOL constants.

6 Future work

In (Robaldo and Sun, 2017), the translation of the whole GDPR into reified Input/Output logic has been advocated as a future work, with the aim of showing that, by simplifying/flattening the architecture of the logical framework through reification, it is possible to build large knowledge bases of formulæ in reasonable time, without limiting the expressivity needed to properly represent knowledge from legal texts.

The present paper provides that evidence; in the future, further evidence may be provided by developing similar knowledge bases encoding norms from other regulations or directives, such as MiFID II[41].

Focusing on the DAPRECO knowledge base and the data protection domain only, lot of further research still needs to be done to make the knowledge base really useful in practical applications, or even only as a benchmark *corpus* of examples for reified Input/Output logic and the LegalRuleML standard.

In particular, we identify two directions for further research, both indispensable to this end: (1) extending the DAPRECO knowledge base with *GDPR operational constraints*, i.e., context-specific requirements detailing how, and to what extent, GDPR norms are met in real-world scenarios, and (2) designing and implementing inference schema to perform legal reasoning tasks on the (extended) knowledge base.

6.1 Extending the DAPRECO knowledge base with GDPR operational constraints

Our future research on the DAPRECO knowledge base is grounded on the distinction between GDPR *formal* requirements (formal compliance) and GDPR *operational* requirements (substantive compliance), how and to what extent the GDPR formal requirements are met in real-world scenarios.

For instance, in Article 12, the GDPR specifies that the controller shall take appropriate measures to provide the data subject with any information about the collection of his personal data or in case of data breach "in a concise, transparent, intelligible and easily accessible form, using clear and plain language". However, the regulation does not detail which measures are appropriate in the different contexts to this end. Rather, it requires controllers to define their own data protection policies (see Articles 13, 14, and 24(2)), whereas associations and other bodies representing categories of controllers or processors are invited to prepare codes of conduct (see Article 40), and the European Data Protection Board has the duty to release guidelines and recommendations (see 70(1)(d)).

At present, the DAPRECO knowledge base represents formal requirements only; to model the example under examination, for instance, it uses a predicate "*clearness*", which is true if, and only if, the fact that the communication is done "in a concise, transparent, intelligible and easily accessible form, using clear and plain language" really exists. The predicate "*clearness*" parallels the same level of vagueness of the original legal text: both the GDPR and the DAPRECO knowledge base do not specify how and to what extent the communication is "clear" enough, in the different domains where

[41] `https://eur-lex.europa.eu/legal-content/IT/ALL/?uri=CELEX:32014L0065&qid=1435045139484`.

personal data are processed (such as Fintech, eHealth, IoT) or with respect to different technologies involved in the processing (such as augmented reality, blockchain).

To this end, in the future the DAPRECO knowledge base needs to be enriched by constitutive rules specifying under which conditions the predicate "*clearness*" is true in the different contexts, possibly with respect to different (and mutually exclusive) legal interpretations. These additional constitutive rules will encode the content of the mentioned recommendations, standards, codes of conduct, and the like, many of which are currently unavailable. subsection 4.3 shows an example with respect to Article 33, paragraph 2, of the GDPR and the control A9.1 of the ISO/IEC 27018:2014 standard. However, this is just an isolated example to show how it is possible to correlate formal and operational requirements, while a more systematic and exhaustive enrichment of the knowledge base in that score is left as a future work.

Most important of all, we believe that the formulæ expressing correlations between GDPR formal and operational constraints must be evaluated and approved by a large set of domain experts with context-specific heterogeneous knowledge and perspectives on GDPR operational constraints; this set includes lawyers and data protection officers, but also security managers, process managers, data managers, and other professionals from academia, industry, and institutions.

Therefore, in our future research we aim at designing and implementing questionnaires to gather feedbacks about the formulæ in the DAPRECO knowledge base while abstracting their formal details. In addition, we aim at making these questionnaires available online to encompass a large number of respondents, similarly to what has been done in (Robaldo, Szymanik, and Meijering, 2014). As the set of documents containing operational requirements is expected to be large, the construction of the questionnaires will be supported by automatic procedures, drawn from past research in NLP (Robaldo et al., 2011; Boella et al., 2013a; Boella et al., 2013b), to help identify new relevant concepts or relate the ones already present in the DAPRECO knowledge base to their matching textual excerpts.

The collected feedback will allow to understand whether, and to what extent, the formal representations reflect the intended meaning of the textual sources, or whether they will rather call for a revision of the formulæ in the DAPRECO knowledge base.

Once a correlation will be supported by a reasonably large set of domain experts, it will be deemed "acceptable", although not "universally valid". It is always possible to add additional legal interpretations, incompatible with the currently acceptable ones, which may be possibly become acceptable in the future.

In other words, the DAPRECO knowledge base must be intended as a "living entity" that will never be exhaustive and correct enough or, at the very most, only for short periods. For this reason, as already pointed out in section 3 above, it is crucial to explicitly represent the temporal dimension via FOL terms such as "t_1", "t_2", to infer that, for instance, organizations that currently implement "appropriate measures" in the sense of Article 12 of the GDPR are not liable for subsequent technological advancements that nullify that level of appropriateness.

Note that approaches based on semi-structured questionnaires are already available in literature to evaluate legal ontologies (Casellas, 2009; Ramakrishna, Gorski, and Paschke, 2016). The PrOnto ontology is currently under evaluation through similar questionnaires, in the context of research projects at the University of Bologna.

On the other hand, no evaluation methodology is available in the literature for fine-grained logical representations such as the ones in the DAPRECO knowledge base, also because these are indeed quite novel. The DAPRECO knowledge base will be the first use case on which developing such a novel evaluation methodology.

6.2 Designing and implementing inference schema to perform legal reasoning tasks on the DAPRECO knowledge base

Using the DAPRECO knowledge base in practical applications requires to define and implement inference schemas for legal reasoning tasks on the if-then rules stored in the knowledge base, first of all automatic compliance checking, which, given a description of the state of affairs, entails the task of determining which GDPR obligations have been fulfilled or violated, as well as which ones are still in force, even in case some other obligations have been already violated (contrary-to-duty reasoning).

The design, and consequent implementation, of such reasoning tasks does not appear to be an easy one, in light of the computational complexity analysis of Input/Output. Satisfiability in Input/Output logic is *coNP* hard and in the second level of the polynomial hierarchy; for instance, contrary-to-duty reasoning is BH_2 complete[42] (see (Sun and Robaldo, 2017), Theorem 3.16).

On the one hand, these complexity results are not so comforting with respect to the goal of using Input/Output logic in practical applications in computer science. However, the main competitors of Input/Output logic face no less problems for standard reasoning tasks. This is particularly true for deontic frameworks grounded on possible-world semantics, which adds an extra machinery that makes the overall computational complexity much worse than the one of Input/Output logic. For instance, STIT logic, one of the main deontic logic grounded on possible-worlds semantics, is undecidable (Schwarzentruber and Semmling, 2014).

Thus, we are still optimistic about the future of the formalism, but we acknowledge that the computational complexity analysis of more advanced Input/Output frameworks, such as the one used for the DAPRECO knowledge base, requires a lot of additional effort.

The basic inference tasks in Input/Output logic are currently limited to the original definitions (Makinson and van der Torre, 2000; Makinson and van der Torre, 2001), which take standard propositional logic as the object logic. Conversely, in the DAPRECO knowledge base, the object logic is the first-order framework in (Hobbs and Gordon, 2017). As explained in subsection 3.1, the formulæ belonging to the assertive contextual statements (the ABox) should be distinguished from those belonging to the terminological declarative statements (the TBox).

The ABox formulæ feature a trivial syntax: they are all conjunctions of atomic predicates, possibly outscoped by existential quantifiers. A pair of such formulæ constitutes an Input/Ouput (if-then) rule. External universal quantifiers bind all variables that are not existentially quantified within the inner formulæ from (Hobbs and Gordon, 2017).

Provided the universe is finite, the if-then rules may be easily reduced to pairs of sets of propositional symbols via Skolemization and enumeration, and the computational

[42] BH_2 is the class of languages which are the intersection of a language in *NP* and a language in *coNP*.

complexity turns out to be trival as the only possible inferences are those allowed by conjunction ('$A, B \vdash A \wedge B$' and '$A \wedge B \vdash A$').

More advanced inferences (such as negation, disjunction, or is-a relations) may be then allowed by introducing corresponding axioms and semantic relations in the reference ontology. It then follows that in reified Input/Output logic the computational complexity wholly depends on the reference ontology; by restricting the expressivity of the TBox axioms, we can then control the complexity of the overall Input/Output system in order, for instance, to make it usable in practical applications.

With respect to the present work, the if-then rules in the DAPRECO knowledge base constitute the ABox of the overall framework, while the TBox is mostly represented by the PrOnto ontology, encoded in `OWL2-DL`.

Therefore, the first future work in this direction we advocate is the extension of the results and proofs in (Sun and Robaldo, 2017) to the expressivity of description logic.

The next step would be to study the computational complexity of the defeasible methods used in the DAPRECO knowledge base to handle assumptions and exceptions (see subsection 4.2 and subsection 4.3 above). Those are respectively assumed to be true and false, but they can be otherwise asserted, in order to block or allow certain inferences.

Note that, although the method to handle defeasibility in reified Input/Output logic is *drawn* from Circumscriptive Logic, the latter is much more complex and technical than just having abnormality predicates (Cadoli and Lenzerini, 1994; Bonatti, Lutz, and Wolter, 2009). In other words, reified Input/Output logic does not convey the whole complexity of the axiom of circumscription.

Our assumptions and exceptions are technically equivalent to standard constructs proposed since the nineties to undercut defeasible rules in logic programming, such as the undercutting defeaters of Defeasible Logic (Nute, 1994a), implemented in defeasible-Prolog programming language (Nute, 1994b). Nowadays, they are widely used in argumentation systems (Amgoud and Nouioua, 2015).

Other works (Governatori et al., 2013) use similar defeaters in a tractable extension. On the other hand, that object logic has a quite reduced expressivity, in that it only allows (modal) literals. Given the simplicity of the object logic, tractability comes out easily. Nevertheless, it appears that defeaters do not add extra complexity *per se*, but the complexity of the overall resulting system again depends only on the inferences allowed by the object logic (the one of the reference ontology), in the reified Input/Output logic setting. A further direction of future work that we advocate is to verify whether this hypothesis is true.

More generally, rather than focusing on the computational complexity analysis of reified Input/Output logic only, a broader and exhaustive future work would be to study how the complexity of the (inner) object logic interacts with the complexity of the (outer) Input/Output wrapper, in order to determine the complexity of the overall resulting framework.

The computational complexity analysis of reasoning tasks on the DAPRECO knowledge base should be then seen as a particular case study of this general future research, but not the single one deserving further investigation.

For instance, (Parent, 2011) proposes an extension of standard Input/Output logic, called prioritized Input/Output logic, suitable to handle moral conflicts; those are resolved based on a priority ordering on the power set of the Input/Output pairs. The complexity of prioritized Input/Output logic is superior to the one of the DAPRECO knowledge base, and its investigation may be seen as a second case study of the general research plan outlined here.

On the other hand, Input/Output logic has not been used for modeling legal reasoning only; for instance, as mentioned in section 2 above, (Bochman, 2004) uses it to model causal reasoning, but the author does not present any computational complexity analysis of its framework either.

To conclude, investigating the computational complexity of the mentioned Input/Output logic frameworks, and others as well, ought to be placed within a general perspective, to avoid duplication of work. The way is still very long, in that the results hold for the propositional level only.

7 Conclusions

In this paper, we have discussed how reified Input/Output logic is a suitable formalism to express complex legal statements as those in the General Data Protection Regulation.

While showing interesting cases, some including defeasible meanings, from the Regulation and from other (exemplified) legal statements, the paper also presented the process of using reified Input/Output logic to express GDPR provisions, thoroughly explaining how the legal statements are translated into formulæ. The result is an extensive data base called the DAPRECO knowledge base.

The current version of the DAPRECO knowledge base includes 966 formulæ in reified Input/Output logic: 271 obligations, 76 permissions, and 619 constitutive rules. To date, the DAPRECO knowledge base is the biggest knowledge base in Input/Output logic and LegalRuleML available online.

The knowledge base 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 portion of the legal text to formalize, build the corresponding formulæ, and save the result in LegalRuleML.

The present paper shows that reified Input/Output logic appears to be a suitable formal instrument to build, in a reasonable time, large knowledge bases of formulæ without limiting the expressivity needed to properly represent content from legal texts.

Reification allows to extend the expressivity of the Input/Output framework (Makinson and van der Torre, 2000; Makinson and van der Torre, 2001) fit to represent GDPR norms and similar deontic linguistic expressions found in existing legislation. On the other hand, as shown in (Robaldo and Sun, 2017), since reification does not affect the model-theoretic semantics of standard Input/Ouput logic, all meta-structures handling contrary-to-duty reasoning, special subtypes of permissions, and the like may be imported in reified Input/Output logic with very little tuning of the formal definitions.

As reified formulæ do not feature nestings of sub-formulæ within complex operators, the proper representation of nested obligations and permissions was achieved quite straightforwardly, by introducing suitable eventualities referring to *the fact that* someone is obliged/permitted to take some actions.

In addition, the avoidance of nestings, and the consequent architectural simplicity of the formulæ, is the factor that allows to build such a large knowledge base 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, Governatori, and Amor, 2017), but no enough exhaustive and systemic work has been conducted so far to translate a whole relevant piece of legislation in LegalRuleML. The DAPRECO knowledge base is the first and unique achievement in this respect.

In our view, the reason is that the formulæ in the underlying logics used in the literature on LegalRuleML, e.g., in (Dimyadi, Governatori, and Amor, 2017), being based on standard embeddings of sub-formulæ within complex operators, are less readable than the ones in reified Input/Output logic, and so harder to edit and debug, in that they require more effort and expertise, especially when the size of the knowledge base increases.

As argued in (Robaldo, 2011), syntactic embeddings are convenient from a computational point of view, in that their model theory may be defined recursively, but they are inadequate for Natural Language Semantics and hard to scale.

On the other hand, to handle exceptions and legal interpretations it was necessary to introduce formal mechanisms to implement defeasibility. The constitutive rules contain special predicates explicitly referring to exceptions and to assumptions taken in legal interpretations. This solution is alternative to others proposed in literature (Rotolo, Governatori, and Sartor, 2015; Parent, 2011), which use priorities or superiority binary operators to infer which rules are “stronger” than the others.

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.

In light of this, it seems there is no need to introduce more complex defeasible schema for modeling legislation. Advanced forms of reasoning could be instead needed whenever we do not know *a priori* which rules override other rules, but this has to be inferred from the asserted knowledge, including the prevalence criteria between different legal sources[43]. Such a reasoning 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.

On the other hand, although reified Input/Output logic introduces predicates explicitly referring to exceptions and assumptions by taking inspiration from Circumscriptive Logic, these are rather limited in their expressivity. Specifically, they correspond to standard defeaters used since decades and still in several modern formalisms, among which the deontic account from (Governatori et al., 2013).

A full and exhaustive computational complexity analysis of Input/Output logic deserves a lot of future work, but is needed to deploy the DAPRECO knowledge base in real-world applications.

[43] Personal communication with Leon van der Torre.

A second important direction of future research that we identified to this end concerns the enrichment of the DAPRECO knowledge base with GDPR *operational constraints*, coming from a plethora of additional documents (guidelines, recommendations, codes of conduct) as specified in several GDPR articles.

The constraints conveyed by these documents specify how GDPR *formal constraints* must be implemented in the different real-world scenarios where personal data are processed. Therefore, they ought to be formalized in reified Input/Output logic as well, linked to the if-then rules stored in the DAPRECO knowledge base, and evaluated by domain experts, *in an incremental and defeasible fashion*. In fact, the if-then rules in the DAPRECO knowledge base will never be exhaustive enough or, at the very most, only for short periods. New official documents, overriding current interpretations, are expected to be released as technology advances. It will be then necessary to incrementally evolve the DAPRECO knowledge base in order to parallel and encompass them.

References

[Ajani et al.2017] Ajani, G., G. Boella, L. Di Caro, L. Robaldo, L. Humphreys, S. Praduroux, P. Rossi, and A. Violato. 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).

[Amgoud and Nouioua2015] Amgoud, Leila and Farid Nouioua. 2015. Undercutting in argumentation systems. In Christoph Beierle and Alex Dekhtyar, editors, *Scalable Uncertainty Management - 9th International Conference, SUM 2015, Québec City, QC, Canada, September 16-18, 2015. Proceedings*, volume 9310 of *Lecture Notes in Computer Science*, pages 267–281. Springer.

[Antoniou et al.2001] Antoniou, Grigoris, David Billington, Guido Governatori, and Michael J. Maher. 2001. Representation results for defeasible logic. *ACM Transactions on Computational Logic*, 2(2):255–287.

[Arner, Barberis, and Buckey2016] Arner, D. W., J. Barberis, and R. P. Buckey. 2016. FinTech, RegTech, and the reconceptualization of financial regulation. *Northwestern Journal of International Law & Business*, 37:371–414.

[Athan et al.2013] Athan, Tara, Harold Boley, Guido Governatori, Monica Palmirani, Adrian Paschke, and Adam Wyner, 2013. *LegalRuleML: From Metamodel to Use Cases*, pages 13–18. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Athan et al.2015] Athan, Tara, Guido Governatori, Monica Palmirani, Adrian Paschke, and Adam Wyner, 2015. *LegalRuleML: Design Principles and Foundations*, pages 151–188. Springer International Publishing.

[Athan et al.2014] Athan, Tara, Guido Governatori, Monica Palmirani, Adrian Paschke, and Adam Z. Wyner. 2014. Legal interpretations in legalruleml. In *Semantic Web for Law 2014 workshop, collocated at the 27th International Conference on Legal Knowledge and Information Systems (JURIX 2014)*, CEUR Workshop Proceedings.

[Bach1981] Bach, E. 1981. On time, tense, and aspect: An essay in english metaphysics. In P. Cole, editor, *Radical Pragmatics*. Academic Press, New York, pages 63–81.

[Bandeira et al.2016] Bandeira, Judson, Ig Ibert Bittencourt, Patrícia Espinheira, and Seiji Isotani. 2016. FOCA: A methodology for ontology evaluation. *CoRR*.

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

[Bochman2004] Bochman, Alexander. 2004. A causal approach to nonmonotonic reasoning. *Artificial Intelligence*, 160(1-2):105–143.

[Boella et al.2016] Boella, G., L. Di Caro, L. Humphreys, L. Robaldo, R. Rossi, and L. van der Torre. 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.

[Boella et al.2012] Boella, G., L. di Caro, L. Humphreys, L. Robaldo, and L. van der Torre. 2012. Nlp challenges for eunomos, a tool to build and manage legal knowledge. Proceedings of the International Conference on Language Resources and Evaluation.

[Boella et al.2013a] Boella, Guido, Luigi Di Caro, Daniele Rispoli, and Livio Robaldo, 2013a. *Semantic Relation Extraction from Legislative Text Using Generalized Syntactic Dependencies and Support Vector Machines*, pages 218–225. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Boella et al.2013b] Boella, Guido, Luigi Di Caro, Daniele Rispoli, and Livio Robaldo. 2013b. A system for classifying multi-label text into eurovoc. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Law*, ICAIL '13, pages 239–240, New York, NY, USA. ACM.

[Boella et al.2010] Boella, Guido, Guido Governatori, Antonino Rotolo, and Leendert van der Torre, 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 and van der Torre2004a] Boella, Guido and Leendert W. N. van der Torre. 2004a. Fulfilling or violating obligations in normative multiagent systems. In *IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT 2004), Beijing, China.*, pages 483–486.

[Boella and van der Torre2004b] Boella, Guido and Leendert W. N. van der Torre. 2004b. 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.

[Bonatti, Lutz, and Wolter2009] Bonatti, Piero A., Carsten Lutz, and Frank Wolter. 2009. The complexity of circumscription in description logic. *Journal of Artificial Intelligence Research*, 35(1):717–773.

[Brank, Grobelnik, and Mladenić2005] Brank, Janez, Marko Grobelnik, and Dunja Mladenić. 2005. A survey of ontology evaluation techniques. In *Proceedings of 8th International multi-conference Information Society*.

[Brozek2014] Brozek, B. 2014. Law and defeasibility. *Revus*, 23:165–170.

[Cadoli and Lenzerini1994] Cadoli, Marco and Maurizio Lenzerini. 1994. The complexity of propositional closed world reasoning and circumscription. *Journal of Computer and System Sciences*, 48(2):255–310.

[Casellas2009] Casellas, Núria. 2009. Ontology evaluation through usability measures. In Robert Meersman, Pilar Herrero, and Tharam Dillon, editors, *On the Move to Meaningful Internet Systems: OTM 2009 Workshops*. Springer Berlin Heidelberg.

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

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

[Dimyadi, Governatori, and Amor2017] Dimyadi, J., G. Governatori, and R. Amor. 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*.

[Galton2006] Galton, Antony. 2006. Operators vs. arguments: The ins and outs of reification. *Synthese*, 150(3):415–441.

[Governatori et al.2013] Governatori, G., F. Olivieri, A. Rotolo, and S. Scannapieco. 2013. Computing strong and weak permissions in defeasible logic. *Journal of Philosophical Logic*, 6(42):799–829.

[Governatori2015] Governatori, Guido. 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.

[Governatori et al.2009] Governatori, Guido, Vineet Padmanabhan, Antonino Rotolo, and Abdul Sattar. 2009. A defeasible logic for modelling policy-based intentions and motivational attitudes. *Logic Journal of the IGPL*, 17(3).

[Governatori and Rotolo2006] Governatori, Guido and Antonino Rotolo. 2006. Logic of violation: a Gentzen system for reasoning with contrary-to-duty obligations. *Australasian Journal of Logic*, (426).

[Governatori and Rotolo2008] Governatori, Guido and Antonino Rotolo. 2008. Bio logical agents: Norms, beliefs, intentions in defeasible logic. *Autonomous Agents and Multi-Agent Systems*, 17(1):36–69.

[Hansen2008] Hansen, Jörg. 2008. Prioritized conditional imperatives: problems and a new proposal. *Autonomous Agents and Multi-Agent Systems*, 17(1):11–35.

[Hansen2014] Hansen, Jörg. 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.

[Hobbs1998] Hobbs, J.R. 1998. The logical notation: Ontological promiscuity. In *Chapter 2 of Discourse and Inference*. Available at http://www.isi.edu/∼hobbs/disinf-tc.html.

[Hobbs2001] Hobbs, J.R. 2001. Syntax and metonymy. In Bouillon P. e Busa F., editor, *The Language of Word Meaning*. Cambridge University Press, pages 302–361.

[Hobbs and Gordon2017] Hobbs, J.R. and A.S. Gordon. 2017. *A formal theory of commonsense psychology, how people think people think*. Cambridge University Press.

[Horty2001] Horty, John. 2001. *Agency and Deontic Logic*. Oxford University Press, New York.

[Horty2012] Horty, John. 2012. *Reasons as Defaults*. Oxford University Press.

[Idelberger et al.2016] Idelberger, Florian, Guido Governatori, Régis Riveret, and Giovanni Sartor. 2016. Evaluation of logic-based smart contracts for blockchain systems. In *RuleML*, volume 9718 of *Lecture Notes in Computer Science*, pages 167–183. Springer.

[Jørgensen1937] Jørgensen, Jorgen. 1937. Imperatives and logic. *Erkenntnis*, 7:288–296.

[MacCormick and Summers1991] MacCormick, N. and R.S. Summers. 1991. *Interpreting Statutes: A Comparative Study*. Applied legal philosophy. Dartmouth.

[Makinson and van der Torre2001] Makinson, David and Leendert van der Torre. 2001. Constraints for input/output logics. *Journal of Philosophical Logic*, 30(2):155–185.

[Makinson and van der Torre2003a] Makinson, David and Leendert van der Torre. 2003a. Permission from an input/output perspective. *Journal of Philosophical Logic*, 32:391–416.

[Makinson and van der Torre2003b] Makinson, David and Leendert van der Torre. 2003b. What is input/output logic? In B. Lowe, W. Malzkorn, and T. Rasch, editors, *Foundations of the Formal Sciences II: Applications of Mathematical Logic in Philosophy and Linguistics*, pages 163–174.

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

[Malerba2017] 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).

[Maranhão2017] Maranhão, Juliano S. A. 2017. A logical architecture for dynamic legal interpretation. In *Proceedings of the 16th Edition of the International Conference on Articial Intelligence and Law*, ICAIL '17, pages 129–138, New York, NY, USA. ACM.

[Maranhão and de Souza2018] Maranhão, Juliano and Edelcio G. de Souza. 2018. Contraction of combined normative sets. In Jan M. Broersen, Cleo Condoravdi, Nair Shyam, and Gabriella Pigozzi, editors, *Deontic Logic and Normative Systems - 14th International Conference, DEON 2018, Utrecht, The Netherlands, July 3-6, 2018.*, pages 247–261. College Publications.

[McCarthy1980] McCarthy, J. 1980. Circumscription: A form of nonmonotonic reasoning. *Artificial Intelligence*, (13):27–39.

[Nute1994a] Nute, D. 1994a. Defeasible logic. In D. Gabbay, C. Hogger, and J. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming*. Oxford University Press.

[Nute1994b] Nute, D. 1994b. Defeasible Prolog. In *AAAI Technical Report FS-93-0, available at https://www.aaai.org/Papers/Symposia/Fall/1993/FS-93-01/FS93-01-015.pdf*. Oxford University Press.

[Nute1997] Nute, D. 1997. *Defeasible Deontic Logic*. Kluwer, Dordrecht.

[Palmirani2011] Palmirani, Monica, 2011. *Legislative Change Management with Akoma-Ntoso*, pages 101–130. Springer Netherlands, Dordrecht.

[Palmirani et al.2018a] Palmirani, Monica, Michele Martoni, Arianna Rossi, Cesare Bartolini, and Livio Robaldo. 2018a. Legal ontology for modelling GDPR concepts and norms. In *Legal Knowledge and Information Systems - JURIX 2018: The Thirty-first Annual Conference, Groningen, The Netherlands, 12-14 December 2018.*

[Palmirani et al.2018b] Palmirani, Monica, Michele Martoni, Arianna Rossi, Cesare Bartolini, and Livio Robaldo. 2018b. Pronto: Privacy ontology for legal compliance. In *Proceedings of the 18th European Conference on Digital Government (ECDG)*, October. Forthcoming.

[Palmirani et al.2018c] Palmirani, Monica, Michele Martoni, Arianna Rossi, Cesare Bartolini, and Livio Robaldo. 2018c. Pronto: Privacy ontology for legal reasoning. In *Proceedings of the 7th International Conference on Electronic Government and the Information Systems Perspective (EGOVIS): Technology-Enabled Innovation for Democracy, Government and Governance*, September. Forthcoming.

[Palmirani et al.2018d] Palmirani, Monica, Michele Martoni, Arianna Rossi, Cesare Bartolini, and Livio Robaldo. 2018d. Pronto: Privacy ontology for legal reasoning. In *Proceedings of the Internationales Rechtsinformatik Symposion (IRIS)*, February.

[Palmirani and Vitali2011] Palmirani, Monica and Fabio Vitali, 2011. *Akoma Ntoso for Legal Documents*, pages 75–100. Springer Netherlands, Dordrecht.

[Parent2011] Parent, Xavier. 2011. Moral particularism in the light of deontic logic. *Artificial Intelligence and Law*, 19(2-3):75–98.

[Parent and van der Torre2014] Parent, Xavier and Leendert van der Torre. 2014. "Sing and dance!". In Fabrizio Cariani, Davide Grossi, Joke Meheus, and Xavier Parent, editors, *Deontic Logic and Normative Systems*, pages 149–165. Springer International Publishing.

[Parent and van der Torre2018] Parent, Xavier and Leendert van der Torre. 2018. Input/output logics with a consistency check. In *Proc. of the 14th International Conference on Deontic Logic and Normative Systems (DEON2018)*.

[Parent and van der Torre2017] Parent, Xavier and Leendert W. N. van der Torre. 2017. The pragmatic oddity in norm-based deontic logics. In *Proc. of the 16th edition of the International Conference on Articial Intelligence and Law, ICAIL 2017, London, United Kingdom, June 12-16, 2017*, pages 169–178.

[Parent and van der Torre2014] Parent, Xavier and Leon van der Torre. 2014. Aggregative deontic detachment for normative reasoning. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Fourteenth International Conference, KR 2014, Vienna, Austria, July 20-24, 2014*.

[Pnueli1977] Pnueli, Amir. 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.

[Prakken2005] Prakken, Henry. 2005. Ai & law, logic and argument schemes. *Argumentation*, 19(3):303–320, Dec.

[Ramakrishna, Gorski, and Paschke2016] Ramakrishna, S., L. Gorski, and A. Paschke. 2016. A dialogue between a lawyer and computer scientist: The evaluation of knowledge transformation from legal text to computer-readable format. *Applied Artificial Intelligence*, 30(3).

[Reiter1987] Reiter, Raymond. 1987. A logic for default reasoning. In M. L. Ginsberg, editor, *Readings in Nonmonotonic Reasoning.* Kaufmann, Los Altos, CA, pages 68–93.

[Robaldo2010a] Robaldo, L. 2010a. Independent set readings and generalized quantifiers. *The Journal of Philosophical Logic*, 39(1):23–58.

[Robaldo2010b] Robaldo, L. 2010b. Interpretation and inference with maximal referential terms. *The Journal of Computer and System Sciences*, 76(5):373–388.

[Robaldo2011] 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.

[Robaldo et al.2011] Robaldo, L., T. Caselli, I. Russo, and M. Grella. 2011. From Italian text to TimeML document via dependency parsing. In *Computational Linguistics and Intelligent Text Processing - 12th International Conference, CICLing 2011, Tokyo, Japan, 2011.*, pages 177–187.

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

[Robaldo, Szymanik, and Meijering2014] Robaldo, L., J. Szymanik, and B. Meijering. 2014. On the identification of quantifiers' witness sets: a study of multi-quantifier sentences. *The Journal of Logic, Language, and Information.*, 23(1).

[Rotolo, Governatori, and Sartor2015] Rotolo, A., G. Governatori, and G. Sartor. 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.

[Sartor2005] Sartor, G. 2005. *Legal Reasoning: A Cognitive Approach to the Law.* Treatise of legal philosophy and general jurisprudence / ed.-in-chief Enrico Pattaro. Springer.

[Satariano2018] Satariano, Adam. 2018. What the G.D.P.R., Europe's tough new data law, means for you, and for the Internet. Online article, May.

[Schwarzentruber and Semmling2014] Schwarzentruber, François and Caroline Semmling. 2014. STIT is dangerously undecidable. In Torsten Schaub, Gerhard Friedrich, and Barry O'Sullivan, editors, *ECAI 2014 - 21st European Conference on Artificial Intelligence, 18-22 August 2014, Prague, Czech Republic - Including Prestigious Applications of Intelligent Systems (PAIS 2014)*, volume 263 of *Frontiers in Artificial Intelligence and Applications.* IOS Press.

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

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

[Sun and Robaldo2015] Sun, Xin and Livio Robaldo. 2015. Logic and games for ethical agents in normative multi-agent systems. In Michael Rovatsos, George A. Vouros, and Vicente Julián, editors, *Multi-Agent Systems and Agreement Technologies - 13th European Conference, EUMAS 2015, and Third International Conference, AT 2015, Athens, Greece, December 17-18, 2015, Revised Selected Papers*, volume 9571 of *Lecture Notes in Computer Science*, pages 367–375. Springer.

[Sun and van der Torre2014] Sun, Xin and Leendert W. N. van der Torre. 2014. Combining constitutive and regulative norms in input/output logic. In Fabrizio Cariani, Davide Grossi, Joke Meheus, and Xavier Parent, 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, Sartor, and Macagno2016] Walton, Douglas, Giovanni Sartor, and Fabrizio Macagno. 2016. An argumentation framework for contested cases of statutory interpretation. *Artif. Intell. Law*, 24(1):51–91.