Combining Input/Output logic and Reification for representing real-world obligations

Livio Robaldo[*], Llio Humphreys[*], Xin Sun[*], Loredana Cupi[+], Cristiana Santos[#], and Robert Muthuri[+]

[*]University of Luxembourg, [+]University of Turin, [#]University of Barcelona
{livio.robaldo, llio.humphreys, xin.sun}@uni.lu, loredana.cupi@unito.it, muthuri.r@gmail.com, cristiana.teixeirasantos@gmail.com
⋆

Abstract. In this paper, we propose a new approach to formalize *real-world* obligations that may be found in existing legislation. Specifically, we propose to formalize real-world obligations by combining insights of two logical frameworks: Input/Output logic, belonging to the literature in deontic logic and normative reasoning, and the Reification-based approach of Jerry R. Hobbs, belonging to the literature in Natural Language Semantics. The present paper represents the first step of the ProLeMAS project, whose main goal is the one of filling the gap between the current logical formalizations of legal text, mostly propositional, and the richness of Natural Language Semantics.

1 Introduction

Legal scholars and practitioners are feeling increasingly overwhelmed with the expanding set of legislation and case law available these days, which is assuming more and more of an international character. Consider, for example, European legislation, which is estimated to be 170,000 pages long, of which over 100,000 pages have been produced in the last ten years.

Legal informatics is an under-researched area in IE, and there is a lack of suitable annotated data. The idiosyncratic nature of legal text poses new challenges for the task of extracting such information using NLP, in order to associate norms with semantic representations on which to perform reasoning [4].

The ProLeMAS project[1] has been specifically proposed to address these challenges. The main goal of the project is to overcome two main limitations of current approaches in normative reasoning and deontic logic:

⋆ Livio Robaldo has received funding from the European Unions Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No 661007 for the project "ProLeMAS: PROcessing LEgal language in normative Multi-Agent Systems". Llio Humphreys is supported by the National Research Fund, Luxembourg. Loredana Cupi has received funding from 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". We would like to thank prof. Leon van der Torre for fruitful discussions.

[1] http://www.liviorobaldo.com/ProLeMAS.htm

(1) a. Several proposals in deontic logic are typically propositional, i.e. their basic components are whole propositions. A proposition basically refers to a whole sentence. On the other hand, natural language (NL) semantics includes a wide range of fine-grained intra-sentence linguistic phenomena: named entities, anaphora, quantifiers, etc. It is then necessary to move beyond the propositional level, i.e. to enhance the expressivity to formalize the meaning of the *phrases* constituting the sentences.

b. Few proposals in deontic logic have been implemented and tested on real legal text. Most of them are only promising methodologies, which overcome the limits of other approaches on the theoretical side. In order to make the logical framework really useful and worth being implemented, its design has to be guided by the analysis of *real* norms.

We started by studying a corpus of EU legislation in English. The corpus includes twenty EU directives from 1998 to 2011, covering a range of subjects, e.g., the profession of lawyer, passenger ships, biotechnological inventions, etc.

Our initial experiments of norm representation in ProLeMAS is conducted on the English version of Directive 98/5/EC of the EU Parliament to facilitate practice of the profession of lawyer on a permanent basis in a Member State other than that in which the qualification was obtained. There are 36 obligations, 13 powers, 10 legal effects, 8 definitions, 6 permissions, 6 applicability types, 6 rationales, 2 rights, 1 exception, and 1 hierarchy. In this paper, we use the following example of obligation for explanatory purposes:

(2) *A lawyer who wishes to practise in a Member State other than that in which he obtained his professional qualification shall register with the competent authority in that State.*

The approach proposed in this paper merges two specific logical frameworks into a new one: (1) Input/Output (I/O) logic, belonging to the literature in deontic logic and normative reasoning, and (2) the Reification-based approach of Jerry R. Hobbs, belonging to the literature in Natural Language Semantics.

I/O logic appears as one of the new achievements in deontic logic in recent years [9]. The key feature of I/O logic is that it adopts *operational* semantics and not truth-conditional ones. Thus, it allows to determine which obligations are operative in a situation that already violates some of them. It is not possible to achieve such a characterization of norms in terms of a truth-conditional semantics: a violation would correspond to an inconsistency, which will make the whole knowledge base inconsistent.

On the other hand, Hobbs's logic is a wide-coverage logic for Natural Language Semantics centered on the notion of Reification. Reification is a concept originally introduced by Donald Davidson in [6]. Modern logical frameworks based on Reification are known in the literature as "neo-Davidsonian" approaches. Reification allows a wide variety of complex natural language (NL) statements to be expressed in First Order Logic (FOL). NL statements are formalized such that events, states, etc., correspond to constants or quantifiable variables of the logic. Following [2], we use in this paper the term 'eventuality' to denote both the reification of a state and the one of an event.

Reification allows us to move from standard FOL notations such as "$(give\ a\ b\ c)$", asserting that "a" gives "b" to "c", to another notation "$(give'\ e\ a\ b\ c)$", which is again in FOL, where e is the *reification* of the giving action. In other words, the expression "$(give'\ e\ a\ b\ c)$" says that "e" is a giving event by "a" of "b" to "c". "e" is a FOL term exactly as "a", "b", and "c".

Many neo-Davidsonian logical frameworks have been proposed in Natural Language Semantics and also in Legal Informatics (cf. section 2 below). The peculiarity of Hobbs's with respect to all other neo-Davidsonian approach is the *total avoidance* of subformulae within the scope of other operators. In other words, the formulae are mere conjunctions of atomic predications. It has been argued in [14] and [25] that many interpretations available in NL require the *parallel* evaluation of two or more logical operators (e.g., modal operators or quantifiers). Section 2 presents some example. Hobbs's logic, by avoiding embeddings of operators within the scope of other operators, straightforwardly and uniformly handles these readings, that are intrinsically prevented in many traditional logical frameworks for Natural Language Semantics.

2 Related work

Some previous approaches try to model, in some deontic settings, sentences coming from *existing norms*. The most representative work is perhaps [30]. Examples of real norms formalized in deontic logic may be also found in [11] and [1].

Many current state-of-the-art approaches try to formalize legal knowledge via Event Calculus [16] [22]. Event Calculus is a neo-Davidsonian logical language that extends the original account of Reification (see [10] for a discussion).

A recent approach in the line is [12]. In [12], it is argued that Event Calculus predicates for handling time cannot be directly used for handling also deontic meaning. Therefore, a new version is proposed to incorporate the deontic effect of norms, so that they can be used for compliance checking. Similar proposals are [24], [7], and [8]. However, [12] appears to be superior in that it identify and formalize much more fine-grained and complex obligation modalities.

The mentioned approaches in Event Calculus focus on business process compliance. In other words, they do not specifically focus on formalizing norms coming from existing legislation.

Thus, they cannot be directly compared with the present proposal, where Natural Language Semantics has a prominent role.

To our knowledge, the approach that appears closest to the one we are going to propose below is perhaps McCarty's Language for Legal Discourse (LLD) [20], [21]. LLD is strongly drawn on previous studies on Natural Language Semantics, it uses Reification, and it has been developed specifically to model real legal text.

An example of McCarthy's Language for Legal Discourse (LLD) is shown in (3). The sentence in (3) is represented via the formula below it.

(3) "The petitioner contends that the regulatory takings claim should not have been decided by the jury"

```
sterm(contends, A,
      [nterm(petitioner, B, [])
       /det(The, nn),
       sterm(decided, C,
             [D,
              aterm(regulatory,E,[F]) &
              nterm(takings,G,[]) &
              nterm(claim,F,[])
              /det(the, nn)])
       && H^pterm(by, H,
                  [C,
                   nterm(jury,I,[])
                   /det(the, nn)])
       /[modal(should),negative,perfect,passive]
```

`sterm`, `nterm`, `aterm`, and `pterm` are reified terms of different kind. For instance, `sterm`s denote reified relations. Thus, the `sterm` on the first line refers to the eventuality denoted by the main verb "contends".

Space constraints forbid us to illustrate all technical details of LLD. We focus only on the two architectural choices most relevant for the present work:

(4) a. Each `*term` is associated with a lexical entry, e.g. "contends", "petitioner", "decided", etc. This is the first argument of the `*term`.
   b. `*term`s may outscope by other `*term`s. E.g., the `sterm` associated with the main verb "contends" outscopes the `nterm` associated with "petitioner" which in turn outscopes the `sterm` associated with "decided".

(4.a-b) make McCarthy's logic very reminiscent of standard representation formalisms used in Natural Language Semantics such as Discourse Representation Theory (DRT) [15] and Minimal Recursion Semantics (MRS) [5].

Nevertheless, [14] and [25] argues that (4.a-b) intrinsically prevents the proper representation of several readings that are actually available in NL utterances. For instance, consider sentences in (5), drawn from the large range of examples considered by Hobbs and Robaldo in their past research in NL semantics.

(5) a. Permission may be obtained, but *it* could take more than one month.
   b. The city does not have a train station, *but* it has a bus station.
   c. If the parents of a student earn *less than 20k* euros per year, then the student is eligible.

Sentence (5.a) highlights that reification can easily give rise to a practical formalism for NL semantics. The pronoun "it" refers to the permission to be obtained. The referent of the pronoun could be then directly identified by the FOL term reifying the eventuality denoted by the main verb of the first clause (see [14] and several other earlier publications by the same author[2]).

(5.b) is an example of concessive relation, one of the trickiest semantic relations occurring in NL: the first clause creates the expectation that the city is unreachable by public transportation. The second clause denies that expectation. A practical way to properly model concessive relations is to reify the eventuality corresponding to the expectation, as proposed in [28]. Note that the expectation is a "hidden" eventuality, i.e., it is not denoted by any lexical item. Thus, (5.b) cannot be represented in LLD via its basic constructs, due to (4.a).

Finally, (5.c) is an example of cumulative reading. The meaning of (5.c) is that if the money *cumulatively* earned by either one of the parents, or by both together[3] is less than 20k, the student is eligible. Cumulative readings have been extensively studied in a reification setting in [26], [27], and [29].

It is quite hard to represent (5.a-c) by embedding operators within the scope of other operators, as it is done in DRT or MRS. For instance, in (5.c) we have two operators/quantifiers: "Two" and "Less than 20k". By embedding the latter within the scope of the former, we get a reading where the student is eligible if *either* parent independently earns less than 20k euros, but the sum of the two earnings is superior to 20k. On the other hand, in order to get the meaning of cumulative readings, the two quantifiers must be evaluated *in parallel*, i.e., none of the two must outscope the other.

This paper defines a reified deontic logic characterized by the *total avoidance* of embeddings in the instantiated formulae, in line with [14] and [29].

The next two sections introduce the formal instruments at the base of our logical formalization: Hobbs's logic and Input/Output logic. The subsequent sections illustrates how the former can be integrated into the latter, while retaining the advantages of both formalisms.

3 Hobbs' logical framework

Jerry R. Hobbs defines a wide-coverage logic for Natural Language Semantics centered on the notion of Reification. Hobbs's logic uses two related kinds of predicates: primed and unprimed. For instance, the predication "$(give\ a\ b\ c)$" seen above is associated with "$(give'\ e\ a\ b\ c)$", where "e" is the reification of the giving action. Hobbs' implements a fairly large set of linguistic and semantic concepts including sets, composite entities, scales, change, causality, time, event structure, etc., into an integrated first order logical formalism.

Eventualities may be *possible* or *actual*. In Hobbs', this distinction is represented via a unary predicate *Rexist* that holds for eventualities really existing in the world. To give an example cited in Hobbs, if I want to fly, my wanting exists in reality, but my flying does not. This is represented as:

[2] See http://www.isi.edu/∼hobbs/csknowledge-references/csknowledge-references.html and http://www.isi.edu/∼hobbs/csk.html.

[3] For instance, suppose they rent an apartment that they co-own.

$$\exists_e[\ (\mathit{Rexist}\ e) \land (\mathit{want}'\ e\ \mathtt{I}\ e_1) \land (\mathit{fly}'\ e_1\ \mathtt{I})\ ]$$

Eventualities can be treated as the objects of human thoughts. Reified eventualities are inserted as parameters of such predicates as *believe*, *think*, *want*, etc. Reification can be applied recursively. The fact that *John believes that Jack wants to eat an ice cream* is represented as an eventuality e such that it holds:

$$\exists_e \exists_{e_1} \exists_{e_2} \exists_{e_3}[\ (\mathit{Rexist}\ e) \land (\mathit{believe}'\ e\ \mathtt{John}\ e_1) \land (\mathit{want}'\ e_1\ \mathtt{Jack}\ e_2) \land \\ (\mathit{eat}'\ e_2\ \mathtt{Jack}\ \mathtt{Ic}) \land (\mathit{iceCream}'\ e_3\ \mathtt{Ic})\ ]$$

Every relation on eventualities, including logical operators, causal and temporal relations, and even tense and aspect, may be reified into another eventuality. For instance, by asserting $(\mathit{imply}'\ e\ e_1\ e_2)$, we reify the implication from e_1 to e_2 into an eventuality e and e is, then, thought of as "the state holding between e_1 and e_2 such that whenever e_1 really exists, e_2 really exists too". Negation is represented as $(\mathit{not}'\ e_1\ e_2)$: e_1 is the eventuality of e_2's not existing.

The predicates *imply*$'$ and *not*$'$ are defined to model the concept of 'inconsistency'. Two eventualities e_1 and e_2 are said to be inconsistent if and only if they (respectively) imply two other eventualities e_3 and e_4 such that e_3 is the negation of e_4. The definition is as follows:

(6) $(\mathit{forall}\ (e_1\ e_2)$
        $(\mathit{iff}\ (\mathit{inconsistent}\ e_1\ e_2)$
            $(\mathit{and}\ (\mathit{eventuality}\ e_1)\ (\mathit{eventuality}\ e_2)$
                $(\mathit{exists}\ (e_3\ e_4)\ (\mathit{and}\ (\mathit{imply}\ e_1\ e_3)$
                                        $(\mathit{imply}\ e_2\ e_4)(\mathit{not'}\ e_3\ e_4))))))$

(6) is an example of an 'axiom schema'. In this logic, an 'axiom *schema*' provides one or more different axioms for each predicate p. The axiom schemas of the predicates used in formulae are generally stored into a separate ontology.

Higher order operators, such as modal and temporal operators, are modelled by introducing new predicates and by defining axiom schemas to restrict their meaning. However, it is important to note that thanks to reification, the formulae representing natural language utterances *never* feature any kind of embedding of predicates within other operators. In other words, formalae are *always* conjunctions of atomic FOL predicates applied to FOL terms. See for instance [28], which propose a formalization in Hobbs' of concessive relations.

As [14], pp.5, states: "There has been an attempt to make the notation as 'flat' as possible. All knowledge is knowledge of predications[4]. FOL terms are only handles. The intuition is that in natural language we cannot communicate entities directly. We can only communicate properties and hope that the listener can determine the entity we are attempting to refer to."

It should be clear that the main peculiarity of Reification-based logical frameworks is their formal simplicity. This eases the handling of several natural language phenomena. Hobbs' past research particularly addresses the proper treatment of anaphora (cf. in particular [14]). For instance, (7.a) may be represented

[4] And, it may be separately asserted in axiom schema.

via formula (7.b). For simplicity, in (7.b) the semantic relation between the two clauses is simply represented as a material implication (predicate *imply′*).

(7) a. If John goes to Mary's house, he tells her before.

b. $(Rexist\ e) \wedge (imply'\ e\ e_1\ e_2) \wedge (goTo'\ e_1\ \mathtt{J}\ \mathtt{M}) \wedge$
$(tell'\ e_2\ \mathtt{J}\ e_1\ \mathtt{M}) \wedge (happenBefore\ e_2\ e_1)$

e_1 is the event of John's going to Mary, while e_2 is the event of John's telling to Mary *the fact that he will come to her*. The predicate $(happenBefore\ e_2\ e_1)$ states that e_2 must occur before e_1. Note that e_1 and e_2 are only hypothetical eventualies: (7.b) does not assert that they exist in the real world. In (7.a), the (hidden) pronoun "it", which is the object of the verb "tell", is straightforwardly represented: the eventuality e_1 is directly inserted as the second parameter of the *tell′* predicate in (7.b). Without Reification, it would be necessary to introduce some 2-order operators in the logic in order to get the intended meaning of (7.a).

Hobbs' formula are formulae in first order logic with a very restricted syntax. They are basically *conjunctions* of atomic predicates instantiated on FOL terms. From a formal point of view, eventualities are FOL terms exactly as "`Jack`" in example (7.b), the only difference being that they refers to facts and actions occurring in the world. Facts and actions are taken to be individuals of the domain like persons, dogs, etc.

The logic we are going to use as the object logic of I/O systems - which we will call "ProLeMAS object logic" - is a further simplification of Hobbs' logic. The formulae will be more verbose, but, in our view, the simpler syntax will enhance readability and it will facilitate the definition of a reference ontology storing the available predicates and the axiom schemas modelling their meaning.

In fact, it is easy to see that the structure of the formulae strictly resemble the technique of rewriting relations of arbitrary arity as binary relations, used in AI as entity-attribute-value (EAV) triples in the last decades, which is at the basis of the subject-predicate-object of the RDF/OWL data model[5].

In ProLeMAS object logic, there is a single type of predicate, i.e. there is no distinction between primed and unprimed predicates. Predicates will be always unary or binary predicates. Thus, for instance, "$(give\ a\ b\ c)$" is not an acceptable predicate in our logic. N-ary relations are modelled by making thematic roles explicit. This is done by introducing other FOL predicates referring each to an available thematic role. For example, "$(give\ a\ b\ c)$" is translated into:

(8) $$(give\ e_1) \wedge (agent\ e_1\ a) \wedge (patient\ e_1\ b) \wedge (recipient\ e_1\ c)$$

The meaning of (8) is obvious: e_1 is a giving event whose agent is a, whose patient is b, and whose recipient is c. "Agent", "patient", and "recipient" are thematic roles. The ontology specifies, for each kind of eventuality, the available thematic roles and - via further axioms - the restrictions on these thematic roles.

[5] http://www.w3.org/TR/owl-ref

For instance, if we want to impose that agents of giving eventualities can be only human beings, we add the following axiom to the ontology:

(9) $(\mathit{forall}\ (e\ a)\ (\mathit{if}\ (\mathit{and}\ (\mathit{give}\ e)\ (\mathit{agent}\ e\ a))\ (\mathit{humanBeing}\ a)))$

Of course, the computational ontology has to be designed and developed with respect to the application and the domain where we want to concretely use the formulae. In our future research, we aim at specifically designing and implementing a legal ontology for modelling the meaning of norms expressed in natural language [3].

We now formally defines[6] the syntax of ProLeMAS object logic. For reasons that will be clear below, ProLeMAS object logic includes existential quantifiers but not universal ones. And, free variables are allowed. Those will be bound by an (external) universal quantifier.

Definition 1 (Syntax of ProLeMAS object logic). *ProLeMAS object logic is a fragment of First Order Logic (FOL). Where:*

- *The vocabulary includes FOL terms (constants, variables, and functions), FOL unary or binary predicates, the boolean connective "$\wedge$" and the existential quantifier "$\exists$".*
- *If "p" and "q" are, respectively, a unary and a binary predicate, while "a" and "a" are terms, "p(a)" and "q(a,b)" are atomic formulae.*
- *If $\Phi_1 \dots \Phi_n$ are atomic formulas, "$\Phi_1 \wedge \dots \wedge \Phi_n$" and "$\exists_{x_1} \dots \exists_{x_m} [\Phi_1 \wedge \dots \wedge \Phi_n]$", where $x_1 \dots x_m$ occurring in $\Phi_1 \dots \Phi_n$, are non-atomic formulas, possibly containing free variables.*

4 Input/Output logic

Input/Output logic (I/O logic) was originally introduced by Makinson and van der Torre in [19]. For a comprehensive survey and a techinqual introduction of I/O logic, see [23] and [31] respectively. Strictly speaking, I/O logic is not a single logic but a family of logics, just like modal logic is a family of logics containing systems K, KD, S4, S5, etc. In the first volume of the handbook of deontic logic and normative systems [9], I/O logic appears as one of the new achievements in deontic logic in recent years.

I/O logic originated from the study of conditional norms. Unlike modal logic, which usually uses possible world semantics, I/O logic mainly adopts operational semantics: an I/O system is conceived as a deductive machine, like a black box which produces deontic statements as output, when we feed it factual statements as input. In the original paper of Makinson and van der Torre, i.e. [19], four I/O logics are defined: $\mathtt{out}_1$, $\mathtt{out}_2$, $\mathtt{out}_3$, and $\mathtt{out}_4$. They vary on the axioms used to constrain the deductive machine that produces the output against a valid input.

[6] Definition 1 only includes the boolean connective "$\wedge$". Other boolean connectives are modelled by introducing special predicates as *imply′* and *not′* in (6).

Let $\mathbb{P} = \{p_0, p_1, \ldots\}$ be a countable set of propositional letters and $\mathtt{L}$ be the propositional language built upon $\mathbb{P}$. Let $\mathtt{G} \subseteq \mathtt{L} \times \mathtt{L}$ be a set of ordered pairs of formulas of $\mathtt{L}$. $\mathtt{G}$ represents the deduction machine of the I/O logic: whenever one of the heads is given in input, the corresponding tails are given in output. Each pair $(a,\, b)$ in $\mathtt{G}$ is called a "generator" and it is read as "given a, it ought to be x". In this paper, "a" and "b" are respectively termed as the *head* and the *tail* of the generator $(a,\, b)$. Formally, $\mathtt{G}$ is a function from $2^{\mathtt{L}}$ to $2^{\mathtt{L}}$ such that for a set $\mathtt{A}$ of formulas, $\mathtt{G}(\mathtt{A}) = \{x \in \mathtt{L} : (a, x) \in \mathtt{G}$ for some $a \in \mathtt{A}\}$. Makison and van der Torre [19] define the semantics of I/O logics from $\mathtt{out}_1$ to $\mathtt{out}_4$ as follows:

(10)
- $\mathtt{out}_1(\mathtt{G},\mathtt{A}) = Cn(\mathtt{G}(Cn(A)))$.
- $\mathtt{out}_2(\mathtt{G},\mathtt{A}) = \bigcap\{Cn(\mathtt{G}(\mathtt{V})) : \mathtt{A} \subseteq \mathtt{V}, \mathtt{V}$ is complete$\}$.
- $\mathtt{out}_3(\mathtt{G},\mathtt{A}) = \bigcap\{Cn(\mathtt{G}(\mathtt{B})) : \mathtt{A} \subseteq \mathtt{B} = Cn(\mathtt{B}) \supseteq \mathtt{G}(\mathtt{B})\}$.
- $\mathtt{out}_4(\mathtt{G},\mathtt{A}) = \bigcap\{Cn(\mathtt{G}(\mathtt{V}) : \mathtt{A} \subseteq \mathtt{V} \supseteq \mathtt{G}(\mathtt{V})), \mathtt{V}$ is complete$\}$.

Here Cn is the classical consequence operator of propositional logic, and a set of formulas is *complete* if it is either *maximal consistent* or equal to $\mathtt{L}$. These four logics are called *simple-minded output*, *basic output*, *simple-minded reusable output* and *basic reusable output* respectively. For each of these four logics, a *throughput* version that allows inputs to reappear as outputs, defined as $\mathtt{out}_i^+(\mathtt{G}, \mathtt{A}) = \mathtt{out}_i(\mathtt{G}_{id}, \mathtt{A})$, where $\mathtt{G}_{id} = \mathtt{G} \cup \{(a, a) \mid a \in \mathtt{L}\}$. When $\mathtt{A}$ is a singleton, we write $\mathtt{out}_i(\mathtt{G}, a)$ for $\mathtt{out}_i(\mathtt{G}, \{a\})$.

I/O logics are given a proof theoretic characterization. We say that an ordered pair of formulas is derivable from a set $\mathtt{G}$ iff (a, x) is in the least set that extends $\mathtt{G} \cup \{(\top, \top)\}$ and is closed under a number of derivation rules. The following[7] are the rules we need to define $\mathtt{out}_1$ to $\mathtt{out}_4{}^+$:

(11)
- SI (strengthening the input): from (a, x) to (b, x) whenever $b \vdash a$.
- OR (disjunction of input): from (a, x) and (b, x) to $(a \vee b, x)$.
- WO (weakening the output): from (a, x) to (a, y) whenever $x \vdash y$.
- AND (conjunction of output): from (a, x) and (a, y) to $(a, x \wedge y)$.
- CT (cumulative transitivity): from (a, x) and $(a \wedge x, y)$ to (a, y).
- ID (identity): from nothing to (a, a).

The derivation system based on the rules SI, WO and AND is called $\mathtt{deriv}_1$. Adding OR to $\mathtt{deriv}_1$ gives $\mathtt{deriv}_2$. Adding CT to $\mathtt{deriv}_1$ gives $\mathtt{deriv}_3$. The five rules together give $\mathtt{deriv}_4$. Adding ID to $\mathtt{deriv}_i$ gives $\mathtt{deriv}_i^+$ for $i \in \{1, 2, 3, 4\}$. $(a, x) \in \mathtt{deriv}_i(\mathtt{G})$ is used to denote the norms (a, x) derivable from $\mathtt{G}$ using rules of derivation system $\mathtt{deriv}_i$. In [19], it is proven that each $\mathtt{deriv}_i^{(+)}$ is sound and complete with respect to $\mathtt{out}_i^{(+)}$.

I/O logic is a general framework for normative reasoning, used to formalize and reason about the detachment of obligations, permissions and institutional facts from conditional norms. I/O logic is not defined in terms of a *truth-conditional* semantics. Rather, as pointed out above, I/O logic adopts *operational* semantics. As explained in [17] and [18], "directives *do not* carry truth-values.

[7] In (11), $\vdash$ is the classical entailment relation of propositional logic.

Only declarative statements may bear truth-values, but norms are items of another kind. They may be respected (or not), and may also be assessed from the standpoint of other norms, for example when a legal norm is judged from a moral point of view (or viceversa). But it makes no sense to describe norms as true or as false."

Thus, I/O logic allows to determine which obligations are operative in a situation that already violates some of them. To achieve this, we must look at the family of all maximal subsets `G'` such that `G'`$\subseteq$`G` and $\texttt{out}_i$(`G'`, `A`) is consistent with `A`. The family of such $\texttt{out}_i$(`G'`, `A`) is called the `outfamily` of (`G`, `A`).

To understand this concept, consider the following example. Suppose we have the following two norms: "The cottage should not have a fence or a dog" and "if it has a dog it must have both a fence and a warning sign." that we may formalize as the I/O logic generators "$(\top, \neg(f \vee d))$" and "$(d, f \wedge w)$" respectively.

Suppose further that we are in the situation that the cottage has a dog. In this context, the first norm is violated. And, the `outfamily` of (`G`, `A`) determines[8] that, still, we are obliged to build a fence with a warning sign around the cottage:

$$\texttt{G} \equiv \{(\top,\ \neg(f \vee d)),\ (d,\ f \wedge w)\},\ \texttt{A} \equiv \{d\},\ \texttt{outfamily(G, A)}\equiv\{Cn(f \wedge w)\}$$

Although Input/Output logic is an adequate framework for representing and reasoning on norms, only propositional logics have been used for asserting the generators and the input so far. This is because of issues related to the complexity of the framework. The complexity of input/output logic is at least as difficucut as the objective logic. By the complexity of input/output logic, we mean the complexity of the following fulfillment problem:

$$\text{Given finite } G, A \text{ and } x, \text{ is } x \in out_i(G, A)?$$

[32] shows that the complexity of the fulfillment problem for $\texttt{out}_1$,$\texttt{out}_2$,$\texttt{out}_4$ is `coNP` complete, for $\texttt{out}_3$ the lower bound is `coNP` while the upper bound is $\texttt{P}^{\texttt{NP}}$.

There have been no efforts to represent norms coming from *existing* legal texts as those from the corpus described above in section 2. For representing concrete existing norms, the expressivity of propositional logic is not sufficient.

First order (object) logics are needed in order to fill the gap between the Input/Output logic and the richness of NL semantics. To this end, in the next section, we propose a merger of the ProLeMAS Object Logic as defined in section **??** with Input/Output logic. There is no precedent in the literature of a first-order Input/Output logic. However, it is worth noticing that this proposal is not the first deontic logic employing first-order relational variables.

5 The ProLeMAS logic

The present section merges together Input/Output logic and the ProLeMAS object logic, whose syntax has been defined above in definition 1. The resulting logic will be termed as "the ProLeMAS logic".

[8] In this example, `outfamily(G, A)`$\equiv\{Cn(f \wedge w)\}$ for all Input/Output logic $\texttt{out}_i^{(+)}$, with i=1,2,3,4.

We are not interested here in proposing first order versions of *all* definitions shown in the previous section, but we will restrict our attention to only those needed for the aims of the ProLeMAS project. In particular, the axiom `OR` in (11) does not appear to be suitable for legal reasoning. To see why consider the following 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". And, suppose John did one of the two, but there is no way to understand *which* one, i.e. if either he killed a dog or he robbed a bank. Logically, John must spend two years in prison. But from a legal reasoning perspective, he must not: only if concrete evidence of what he did is found, obligations apply. Thus, in the rest of the paper we will no longer consider the `OR` axiom and, consequently, the Input/Output logic $\mathtt{out}_2$ and $\mathtt{out}_4$. On the other hand, we will focus in particular on the `CT` (cumulative transitivity) axiom, used to define the *simple-minded reusable output* logic $\mathtt{out}_3$. It will also be easy to apply the considerations below to the remaining axioms `SI`, `WO`, `AND`, and `ID`, so that we will skip formal definitions about them.

From a formal point of view, recalling that the syntax of the ProLeMAS object logic admits free variables, the last ingredient needed to merge ProLeMAS object logic and $\mathtt{out}_3$ are quantifiers bounding each a free variable. We impose these free variables to occur both in the head and the tail of a generator, and we bound them via universal quantifiers. This establishes a "bridge" between the head and the tail, needed to "carry" individuals from the input to the output. Consider these simple (toy) examples:

(12) a. Each lawyer *must* run.
  b. A lawyer who runs *must* wear a pair of shoes.
  c. If John goes to Mary's house, he'll *have to* tell her before.

We propose to represent (12.a-c) via the following generators:

(13) a. $\forall_x(\ lawyer(x),\ \exists_{e_r}[(Rexist\ e_r) \wedge run(e_r) \wedge agent(e_r,\ x)]\ )$

  b. $\forall_x\forall_{e_r}(\ lawyer(x) \wedge (Rexist\ e_r) \wedge run(e_r) \wedge agent(e_r,\ x),$
  $\exists_{e_w}\exists_y[(Rexist\ e_w) \wedge wear(e_w) \wedge agent(e_w,\ x) \wedge$
  $patient(e_w,\ y) \wedge shoes(y)]\ )$

  c. $\forall_{e_g}(\ (Rexist\ e_g) \wedge go(e_g) \wedge agent(e_g,\ \mathtt{J}) \wedge to(e_g,\ \mathtt{M}),$
  $\exists_{e_t}[(Rexist\ e_t) \wedge tell(e_t) \wedge agent(e_t,\ \mathtt{J}) \wedge receiver(e_t,\ \mathtt{M}) \wedge$
  $theme(e_t,\ e_g) \wedge (happenBefore\ e_t\ e_g)]\ )$

Note that, in (13.b), x occurs in *both* the head *and* the tail of the generator, while e_r *only* occurs in the head. On the other hand, y and e_w are existentially quantified variables that occur *only* in the tail: every time a lawyer runs, there is a different "wearing" eventuality and (possibly) a different pair of shoes.

On the other hand, in (13.c), the eventuality e_g occurs in both the head and the tail. That's because the sentence means: "If John goes to Mary's house, he'll have to tell *Mary* before *that he goes to Mary*".

In our solution, free variables occurring in the heads (and possibly *also* in the tails) are outscoped by universal quantifiers. Free variables occurring *only* in the tails are outscoped by the existential quantifiers of the ProLeMAS object logic syntax (cf. definition 1). Formally:

Definition 2 (ProLeMAS logic generators).
A generator in ProLeMAS logic is a construct in the form:

$$\forall_{x_1,\ldots,x_n,y_1,\ldots,y_m}(\ \Phi(x_1,\ldots,x_n,\,y_1,\ldots,y_m),\,\Psi(y_1,\ldots,y_m)\ )\in \mathtt{G}$$

where $x_1,\ldots,x_n$ are free variables occurring only in Φ while $y_1,\ldots,y_n$ occur both in Φ and in Ψ. Φ and Ψ do not contain any other free variable. Furthermore, Φ does not contain existential quantifiers.

Sentence (2), copied in (14) for reader's convenience, which come from an EU directive in our corpus, can be represented in ProLeMAS logic in a straightforward manner. We simply increase the *size* of the formula, but not its *complexity.*

(14) *A lawyer who wishes to practise in a Member State other than that in which he obtained his professional qualification shall register with the competent authority in that State.*

The formula is:

$$\forall_x\forall_y\forall_{e_w}\forall_{e_p}(\ cond(x,\,y,\,e_w,\,e_p),\,action(x,\,y)\ )$$

where:

$$\begin{aligned} cond(x,\,y,\,e_w,\,e_p)\Leftrightarrow\ & lawyer(x)\wedge memberState(y)\wedge different(y,\,f_w(x))\wedge\\ & Rexist(e_w)\wedge want(e_w)\wedge practise(e_p)\wedge agent(e_w,\,x)\wedge\\ & patient(e_w,\,e_p)\wedge agent(e_p,\,x)\wedge at(e_p,\,y) \end{aligned}$$

$$\begin{aligned} action(x,\,y)\Leftrightarrow \exists_{e_r}[\ & Rexist(e_r)\wedge register(e_r)\wedge agent(e_r,\,x)\wedge\\ & patient(e_r,\,x)\wedge with(e_r,\,f_c(y))\ ] \end{aligned}$$

where x, y, e_w, e_p, and e_r are variables denoting a lawyer, a Member State, and the eventualities of “wanting”, “practising” and “registering”. *agent*, *patient*, *at*, and *with* are thematic roles of the two eventualities. $f_w(x)$ is a function referring to the Member State where x obtained his professional qualification while $f_c(y)$ is a function that given a member state y returns the competent authority of y. The meaning of the formula is obvious: for every tuple of lawyer, Member State, and “wanting”, and “practising” actions that satisfy together the predicates in *cond*, the predicates in *action* are instantiated on x and y.

6 Working with ProLeMAS formulae

The main peculiarity of Reification-based logical frameworks is their formal simplicity. By instantiating FOL predicates on non-variable FOL terms, we obtain again propositional formulae. Thus, it is easy to see that ProLeMAS's generators do not increase the complexity of the Input/Output logic originally defined in [19], provided that two requirements are met: (1) the domain is finite; and (2) the input formulae are only atomic formulae in ProLeMAS object logic, i.e, FOL predicates instantiated on non-variable FOL terms, i.e., propositional formulae.

The aim of the ProLeMAS project is to build concrete NLP-based applications to be used in practical applications, where both requirements (1) and (2) are met. The domains of individuals will always be finite (e.g., the set of all lawyers in the EU). And, we are interested in performing normative reasoning on specific[9] individuals only, e.g., deriving all obligations a specific lawyer must obey, according to a specific normative code.

Universal quantifiers are just a compact way to refer to all individuals in the domain. We obtain equivalent formulae by simply substituing the universally quantified variables with all constants referring each to an individual of the domain. For instance, assume `G` contains generator (13.a) only:

$$\texttt{G} = \{\forall_x(\ lawyer(x),\ \exists_{e_r}[(Rexist\ e_r) \wedge run(e_r) \wedge agent(e_r,\ x)]\ )\}$$

And, suppose the domain is made up of the individuals `John` and `Jack`, the former being a lawyer, the latter not. `G` is equivalent to the following `G'`:

$$\begin{aligned}\texttt{G'}=\{\ &(\ lawyer(\texttt{John}),\ \exists_{e_r}[(Rexist\ e_r) \wedge run(e_r) \wedge agent(e_r,\ \texttt{John})]\ )\\ &(\ lawyer(\texttt{Jack}),\ \exists_{e_r}[(Rexist\ e_r) \wedge run(e_r) \wedge agent(e_r,\ \texttt{Jack})]\ )\ \}\end{aligned}$$

Since `John` is a laywer while `Jack` is not, the *propositional* symbol "$lawyer(\texttt{John})$" belongs to our initial facts while "$lawyer(\texttt{Jack})$" does not, i.e. $lawyer(\texttt{John})\in\texttt{A}$. In $\texttt{out}_1$, we infer that `John` is obliged to run, i.e.

$$\texttt{out}_1(\texttt{G'},\ \texttt{A})=\{\ run(f_1(\texttt{John})) \wedge agent(f_1(\texttt{John}),\ \texttt{John})\ \}$$

Where we substituted[10] the existential quantifier on e_r with a Skolem function f_1, so that $\texttt{out}_1(\texttt{G'},\ \texttt{A})$ again contains propositional symbols only. Thus, it satisfies requirement (2) and, in $\texttt{out}_3$, it could be reused to trigger other obligations, e.g., being applied to an obligation such as "every runner must wear a pair of shoes".

[9] It could be the case that such applications will have to reason on *sets*, e.g., a sets of ten laywers. To properly deal with sets, Hobbs introduces in his framework the notion of *typical element* (cf. [13] and [14]).

[10] Skolemization is merely a formality to meet requirement (2). Alternatively, we could allow existential quantifiers on inputs and define a different pattern-matching rule between the input and the heads of the generators.

We stress again that, thanks to Reification, we are not increasing the complexity of the original I/O logic. On finite domains, the formulae we are going to use turns out to be propositional. Original I/O logic definitions and proofs of soundness and completeness still hold, modulo generalizations via universal and existential quantifiers. For instance, CT is modified as follows:

$$\begin{gathered}
\texttt{CT}\text{ (cumulative transitivity):}\\
\text{from}\\
\forall_{x_1..x_n}(\ \Phi(x_1,\ldots,x_n),\ \exists_{z_1..z_k}[\Psi(y_1,\ldots,y_m,z_1,\ldots,z_k)]\ ),\\
\text{with } \{y_1,\ldots,y_m\}\subseteq\{x_1,\ldots,x_n\}\\
\text{and}\\
\forall_{x_1..x_n w_1..w_r}(\ \Phi(x_1,\ldots,x_n)\wedge\Psi(w_1,\ldots,w_r),\ \exists_{k_1..k_i}[\Upsilon(t_1,\ldots,t_l,k_1,\ldots,k_i)]\ ),\\
\text{with } \{t_1,\ldots,t_l\}\subseteq\{x_1,\ldots,x_n,w_1,\ldots,w_r\}\\
\text{to}\\
\forall_{x_1..x_n}(\ \Phi(x_1,\ldots,x_n),\ \exists_{k_1..k_i m_1..m_s}[\Upsilon(p_1,\ldots,p_c,k_1,\ldots,k_i,m_1,\ldots,m_s)]\ ),\\
\text{with } \{m_1,\ldots,m_s\}\subseteq\{w_1,\ldots,w_r\} \text{ and } \{m_1,\ldots,m_s\}\cup\{p_1,\ldots,p_c\}\equiv\{t_1,\ldots,t_l\}
\end{gathered}$$

7 Future Work and Conclusions

This paper is part of the ProLeMAS research project, which aims at (1) filling the gap between the current logical formalizations of legal text, mostly propositional, and the richness of Natural Language Semantics, and (2) formalizing norms extracted from existing legal documents.

The first step is to move beyond the propositional level towards first-order logical frameworks, in order to enhance the expressivity fit to formalize the meaning of the phrases constituting the sentences. ProLeMAS proposes to achieve such a result by using the Reification-based constructs from Hobbs. The key feature of Hobbs's approach, which distinguishes it from other neo-Davidsonian approaches, is the total avoidance of embeddings of logical operators within the scope of other logical operators.

This paper shows how it is possible to integrate Hobbs's account within I/O logic by adding universal and existential quantifiers to the latter. It also discusses how, provided the domain is finite, the complexity of the resulting framework does not increase with respect to that of propositional I/O logic. We consider this a great result, due to the complexity issues related to I/O logic.

Our next steps will involve the following future work:

(15) a. Studying how the ProLeMAS logic could deal with other kinds of norms, such as permissions, powers, etc. And, eventually, extending the account to provide a proper representation of their meaning.

b. Designing suitable legal ontologies to represent and restrict the meaning of relevant predicates, in order to trigger automatic reasoning on the individuals in the Abox. We are particularly interested in developing legal ontologies in the data protection domain. Under the pressure from technological developments during the last few years, the EU legislation on data protection has shown its weaknesses, and is currently undergoing a long and complex reform that is finally approaching completion.

c. Building a concrete pipeline to populate the Abox of the ontology. In ProLeMAS, the pipeline will firstly process the documents via dependency parsing, then it will define a syntax-semantic interface from the dependency trees to the final formulae.

References

1. M. Araszkiewicz and K. Pleszka, editors. *Logic in the theory and practice of law-making.* Springer, 2015.
2. E. Bach. On time, tense, and aspect: An essay in english metaphysics. In P. Cole, editor, *Radical Pragmatics*, pages 63–81. Academic Press, New York, 1981.
3. Cesare Bartolini and Robert Muthuri. Reconciling data protection rights and obligations: An ontology of the forthcoming eu regulation. In *(To appear in) Proceedings of the Workshop on Language and Semantic Technology for Legal Domain (LST4LD)*, 2015.
4. Guido Boella, Luigi Di Caro, Alice Ruggeri, and Livio Robaldo. Learning from syntax generalizations for automatic semantic annotation. *Journal of Intelligent Information Systems*, 43(2):231–246, 2014.
5. A. Copestake, D. Flickinger, and I.A. Sag. Minimal Recursion Semantics. An introduction. *Journal of Research on Language and Computation.*, 2(3), 2005.
6. D. Davidson. The logical form of action sentences. In Nicholas Rescher, editor, *The Logic of Decision and Action.* University of Pittsburgh Press, 1967.
7. D. Evans and D. Eyers. Deontic logic for modelling data flow and use compliance. In *Proceedings of the 6th International Workshop on Middleware for Pervasive and Ad-hoc Computing*, pages 19–24, New York, NY, USA, 2008. ACM.
8. N. Fornara and M. Colombetti. Specifying artificial institutions in the event calculus. In *V. Dignum editor, Handbook of Research on Multi-Agent Systems: Semantics and Dynamics of Organizational Models*, pages 335–366. IGI Global, 2009.
9. D. Gabbay, J. Horty, X. Parent, R. van der Meyden, and L. (eds.) van der Torre. *Handbook of Deontic Logic and Normative Systems.* College Publications, 2013.
10. Antony Galton. Operators vs. arguments: The ins and outs of reification. *Synthese*, 150(3):415–441, 2006.
11. G. Governatori, F. Olivieri, A. Rotolo, and S. Scannapieco. Computing strong and weak permissions in defeasible logic. *Journal of Philosophical Logic*, 6(42), 2013.
12. M. Hashmi, G. Governatori, and M. Wynn. Modeling obligations with event-calculus. In Antonis Bikakis, Paul Fodor, and Dumitru Roman, editors, *Rules on the Web. From Theory to Applications*, volume 8620 of *Lecture Notes in Computer Science*, pages 296–310. Springer International Publishing, 2014.
13. J.R. Hobbs. Monotone decreasing quantifiers in a scope-free logical form. In *in Semantic Ambiguity and Underspecification*, pages 55–76. 1995.

14. J.R. Hobbs. The logical notation: Ontological promiscuity. In *Chapter 2 of Discourse and Inference*. 1998. http://www.isi.edu/∼hobbs/disinf-tc.html.
15. Hans Kamp and Uwe Reyle. *From Discourse to Logic: an introduction to modeltheoretic semantics, formal logic and Discourse Representation Theory*. Kluwer Academic Publishers, Dordrecht, 1993.
16. R Kowalski and M Sergot. A logic-based calculus of events. *New Generation Computing*, 4(1):67–95, 1986.
17. David Makinson and Leendert van der Torre. Permission from an input/output perspective. *Journal of Philosophical Logic*, 32(4):391–416, 2003.
18. David Makinson and Leendert van der Torre. What is input/output logic? In Benedikt Lwe, Wolfgang Malzkom, and Thoralf Rsch, editors, *Foundations of the Formal Sciences II*, volume 17 of *Trends in Logic*, pages 163–174. Springer Netherlands, 2003.
19. David Makinson and Leendert W. N. van der Torre. Input/output logics. *Journal of Philosophical Logic*, 29(4):383–408, 2000.
20. L. T. McCarty. A language for legal discourse i. basic features. In *Proc. of the 2nd International Conference on Artificial Intelligence and Law (ICAIL '89)*, ACM Press, 1989.
21. L. T. McCarty. Deep semantic interpretations of legal texts. In *The Eleventh International Conference on Artificial Intelligence and Law, Proceedings of the Conference, June 4-8, 2007, Stanford Law School, Stanford, California, USA*, pages 217–224, 2007.
22. R. Miller and M. Shanahan. The event calculus in classical logicalternative axiomatizations. *Electronic Transactions on Artificial Intelligence*, 16(4), 1999.
23. X. Parent and L. van der Torre. Input/output logic. In J. Horty, D. Gabbay, X. Parent, R. van der Meyden, and L. van der Torre, editors, *Handbook of Deontic Logic and Normative Systems*. College Publications, London, 2013.
24. A. Paschke and M. Bichler. SLA representation, management and enforcement. In *2005 IEEE International Conference on e-Technology, e-Commerce, and e-Services (EEE 2005), 29 March - 1 April 2005, Hong Kong, China*, pages 158–163, 2005.
25. L. Robaldo. Interpretation and inference with maximal referential terms. *The Journal of Computer and System Sciences*, 76(5):373–388, 2010.
26. L. Robaldo. Distributivity, collectivity, and cumulativity in terms of (in)dependence and maximality. *The Journal of Logic, Language, and Information*, 20(2):233–271, 2011.
27. L. Robaldo. Conservativity: a necessary property for the maximization of witness sets. *The Logic Journal of the IGPL*, 21(5):853–878, 2013.
28. L. Robaldo and E. Miltsakaki. Corpus-driven semantics of concession: Where do expectations come from? *Dialogue&Discourse*, 5(1), 2014.
29. L. Robaldo, J Szymanik, and B. Meijering. On the identification of quantifiers' witness sets: a study of multi-quantifier sentences. *The Journal of Logic, Language, and Information.*, 23(1), 2014.
30. M. J. Sergot, F. Sadri, R. A. Kowalski, F. Kriwaczek, P. Hammond, and H. T. Cory. The british nationality act as a logic program. *Commun. ACM*, 29(5):370–386, 1986.
31. Xin Sun. How to build input/output logic. In *15th International Workshop on Computational Logic in Multi-Agent Systems*, pages 123–137, 2014.
32. Xin Sun and Diego Agustin Ambrossio. On the complexity of input/output logic. In *to appear in the Proceedings of The Fifth International Conference on Logic, Rationality and Interaction*, 2015.