Towards compliance checking in reified I/O logic via SHACL

Livio Robaldo
livio.robaldo@swansea.ac.uk
Legal Innovation Lab Wales - Swansea University
Swansea, Wales, UK

ABSTRACT

Reified Input/Output logic [29] has been recently proposed to handle natural language meaning in Input/Output logic [17]. So far, the research in reified I/O logic has focused only on KR issues, specifically on how to use the formalism for representing contextual meaning of norms (see [28]). This paper is the first attempt to investigate reasoning in reified I/O logic, specifically compliance checking. This paper investigates how to model reified I/O logic formulae in Shapes Constraint Language (SHACL) [2], a recent W3C recommendation for validating and reasoning with RDFs/OWL.

KEYWORDS

reified I/O logic, SHACL, RDFs/OWL

1 INTRODUCTION

Reified Input/Output logic [29] is Input/Output logic [17] enriched with *reification*. The introduction of reification in I/O logic enhances the expressivity of the I/O logic formulae without substantially affecting the I/O logic constructs that implement deontic reasoning.

Reification is a formal mechanism that associates instantiations of high-order predicates and operators with FOL terms [13], [27], [26]. The latter can be then directly inserted as arguments of other FOL predicates, which may be in turn reified again into new FOL terms. In other words, reified I/O logic associates norms with explicit *terms*, e.g., constants or variables, and not only with truth-conditional symbols such as predicates or (second-order) deontic operators. These terms can be then inserted as parameter of separated meta-properties.

Reified I/O logic is grounded on a specific reification-based approach for Natural Language Semantics: the framework in [12]. The main insight of [12] is to *massively* use reification in order to transform *every* second-order operator, including boolean connectives, into a FOL predicate applied to FOL terms. The final resulting formulae are then *flat conjunctions* of atomic FOL predicates.

As shown in [28], the formal simplicity and the modular structure of reified I/O logic facilitate the implementation of user-friendly interfaces to encode large knowledge bases of norms in reasonable time. [28] presents the DAPRECO knowledge base (D-KB), a repository of 966 formulae in reified I/O formulae that translates norms from the GDPR. The D-KB was built in four months via a special JavaScript editor implemented to this purpose.

While past research in reified I/O logic has focused on how *building* formulae associated with norms in natural language, this paper represents the first attempt to investigate how these formulae can be *implemented and used* for compliance checking, i.e., to infer which obligations have been violated in a given state of affairs and with respect to a given set of norms.

Compliance checking has never been really studied in I/O logic. Most past literature in I/O logic has focused on deontic reasoning, and, recently, normative reasoning [15].

Deontic reasoning is to reason about what is obligatory and permitted, while dealing with contrary-to-duty reasoning, deontic paradoxes, ethical/moral conflicts, etc. Reasoning about obligations and permissions is of course orthogonal to what agents really do, i.e., whether they did or did not violate their obligations or whether they did or did not perform what they were permitted to do.

Compliance checking does not involve deontic reasoning. Still, compliance checking could not be so simple to handle, e.g., because norms might include exceptions that lead to *defeasible* reasoning.

This paper proposes a formalization of non-deontic inferences in reified I/O logic via SHACL [2]. While recent literature offered solutions for compliance checking implemented in RDFs/OWL, e.g., [6], only preliminary works use SHACL to this end, e.g., [21].

2 BACKGROUND - REIFIED I/O LOGIC

2.1 Input/Output logic

I/O logic was originally introduced in [17]. I/O logic is a family of logics, just like modal logic is a family of systems K, S4, S5, etc.

However, while modal logic uses *possible world* semantics, I/O logic uses *norm-based* semantics, in the sense of [11]: I/O systems are families of if-then rules (a, b), such that when a is given in input, b is returned in output. a and b are formulae in another logic, called "the object logic". It has been argued that norm-based reasoning features some advantages over reasoning based on possible world semantics, first of all a lower computational complexity [30].

I/O logic neatly decouples deontic and non-deontic inferences. I/O logic is indeed a *meta-logic* wrapped around another logic (e.g., [12], in case of reified I/O logic) called "the object logic". The meta-logic implements deontic inferences while the object logic implements the non-deontic ones. In I/O systems for legal reasoning, rules (a, b) can be obligations, permissions, and constitutive rules. These are clustered within three distinct sets O, P, and C such that $\forall(a, b) \in O$ reads as "*given a, b is obligatory*", $\forall(a, b) \in P$ reads as "*given a, b is permitted*", and $\forall(a, b) \in C$ reads as "*given a, b holds*".

Most past research on I/O logic has focused on theoretical investigations in the meta-logic, for modeling deontic reasoning. Since the focus was on studying the meta-logic, the object logic was always kept as simple as possible, i.e., it was always *propositional logic*. Reified I/O logic is perhaps the most relevant proposal so far in the I/O logic literature that employs an alternative (first-order) object logic: the logical framework in [12].

In I/O logic, inferences in the meta-logic are achieved by imposing axioms and constraints on the sets of if-then rules. Different combinations of axioms and constraints trigger different inferences.

For instance, [17] defines the basic axioms in (1), where the symbol '$\vdash$' is the entailment relation of the object logic. Variants of these axioms have been further investigated in [23] and [22].

(1)
- SI: from (a, x) to (b, x) whenever $b \vdash a$.
- OR: from (a, x) and (b, x) to $(a \vee b, x)$.
- WO: from (a, x) to (a, y) whenever $x \vdash y$.
- AND: from (a, x) and (a, y) to $(a, x \wedge y)$.
- CT: from (a, x) and $(a \wedge x, y)$ to (a, y).

By imposing axioms SI, WO, and AND, we obtain a specific derivation system called $\texttt{deriv}_1$. Adding OR to $\texttt{deriv}_1$ gives $\texttt{deriv}_2$. Adding CT to $\texttt{deriv}_1$ gives $\texttt{deriv}_3$. The five axioms together give $\texttt{deriv}_4$. Each derivation system is sound and complete with respect to a different (norm-based) semantics and can therefore trigger different inferences (see [17] for further discussion and details).

Given a derivation system, we may further constrain its sets of if-then rules, by considering only subsets that do not yield outputs conflicting with given inputs. This is needed to handle contrary-to-duty reasoning, i.e., to determine which obligations are detached in a situation that already violates some among them [16].

This paper is not concerned with the meta-level of I/O logic. Rather, it will focus on the object logic and non-deontic inferences, including defeasible ones to handle exceptions in legal reasoning.

2.2 Adding reification to I/O logic

Reification is a well-known technique used in linguistics and computer science for representing *abstract* concepts. These are associated with explicit objects, e.g., FOL terms (see below in this section) or RDF resources (see §3 below), on which we can assert properties. These assertions can be recursively reified again into new terms.

Both [12] and RDFs/OWL recursively reify assertions until the knowledge is represented in terms of a *flat* list of atomic predicates applied to terms. In RDFs/OWL, these flat lists are made of triples "`(subject, predicate, object)`", while [12] also allows predicates with higher arity; however, any n-ary predicate can be transformed into an equivalent conjunction of RDF triples.

In [12] and in reified I/O logic, both the antecedent and the consequent of the if-then rule are conjunctions of predicates. Universal and existential quantifiers are added to bound the free variables occurring in the formulae. Universals that outscope the whole if-then rules are used to "carry" individuals from the antecedent to the consequent. Formal details and definitions are available in [29].

A simple example from the D-KB [28] is shown in (2). (2) encodes in reified I/O logic part of Art.5(1)(a) of the GDPR. The if-then rule belongs to the set O (note "$\in O$" in (2)): it is an obligation requiring each personal data processing to be *lawful*.

(2) $\forall_{e_p}(\exists_{t_1,z,w,y,x}[\ (RexistAtTime\ e_p\ t_1) \wedge$
$(PersonalData\ z\ w) \wedge (DataSubject\ w) \wedge$
$(Controller\ y\ z) \wedge (Processor\ x) \wedge (nominates\ y\ x) \wedge$
$(PersonalDataProcessing'\ e_p\ x\ z)\ ],$
$(isLawful\ e_p)\ ) \in O$

Formulae in reified I/O logic employ two kind of predicates: primed predicates such as *PersonalDataProcessing*' and non-primed predicates such as *DataSubject*. The former are obtained by reifying the latter; the first argument of primed predicates is the reification of the non-primed counterpart, i.e., a FOL term.

We should not reify all predicates, but only those we need. For instance, we do need to reify $(PersonalDataProcessing\ x\ z)$ into $(PersonalDataProcessing'\ e_p\ x\ z)$, where e_p explicitly refers to the action of processing, because *we need to assert a property on this action*: in the consequent of the obligation, we require it to be lawful, i.e., to satisfy the *isLawful* predicate. Note that in (2), in order to "carry" the variable e_p from the antecedent to the consequent, a universal quantifier outscoping the if-then rule has been inserted. All other variables are existentially quantified within the antecedent.

The other predicate that e_p is required to satisfy is *RexistAtTime*. This is a special predicates used to assert which reifications "really exist" at a certain time. *RexistAtTime* parallels the well-known predicate *HoldsAt* used in Event Calculus [14].

Thus, formula (2) reads: "for every personal data processing e_p of some personal data z, owned by a data subject w, controlled by a controller y, and processed by a processor x (nominated by y), it is obligatory for e_p to be lawful.

2.3 Adding defeasibility to reified I/O logic

It is common in legislation that some rules override others in restricted contexts. These more specific rules are seen as exceptions of the general rules, as penguins may be seen as exceptions of birds with respect to the ability of flying.

In line with the literature, e.g., [10], reified I/O logic models exceptions via special predicates "Ex" that are false by default. This is achieved via negation-as-failure (naf). "naf(Ex)" is true if either "Ex" is false or it is *unknown*. On the other hand, when "Ex" holds, "naf(Ex)" is false, and the general rule is blocked. An example, taken from [28], is given by the following rules:

(a) If the data subject has given consent to processing, then the processing is lawful.
(b) If the age of the data subject is lower than the minimal age for consent of his member state, (a) is not valid.
(c) In case of (b), if the holder of parental responsibility has given consent to processing, then the processing is lawful.

(a)-(c) are formalized as the following constitutive rules:

(3) $\forall_{e_p}(\exists_{t,z,w,y,x}[\ (RexistAtTime\ e_p\ t) \wedge (DataSubject\ w) \wedge$
$(PersonalDataProcessing'\ e_p\ x\ z) \wedge (Controller\ y\ z) \wedge$
$(Processor\ x) \wedge (nominates\ y\ x) \wedge (PersonalData\ z\ w) \wedge$
$(GiveConsentTo\ w\ e_p) \wedge naf((exceptionAgeDS\ e_p))\ ],$
$(isLawful\ e_p)\ ) \in C$

(4) $\forall_{e_p}(\exists_{t,z,w,y,x,s}[\ (RexistAtTime\ e_p\ t) \wedge (DataSubject\ w) \wedge$
$(PersonalDataProcessing'\ e_p\ x\ z) \wedge (Controller\ y\ z) \wedge$
$(Processor\ x) \wedge (nominates\ y\ x) \wedge (PersonalData\ z\ w) \wedge$
$(StateOf\ s\ w) \wedge (<\ ageOf(w)\ \ minConsentAgeOf(s))\ ],$
$(exceptionAgeDS\ e_p)\ ) \in C$

(5) $\forall_{e_p}(\exists_{t,z,w,y,x,s,h}[\ (RexistAtTime\ e_p\ t) \wedge (DataSubject\ w) \wedge$
$(PersonalDataProcessing'\ e_p\ x\ z) \wedge (Controller\ y\ z) \wedge$
$(Processor\ x) \wedge (nominates\ y\ x) \wedge (PersonalData\ z\ w) \wedge$
$(StateOf\ s\ w) \wedge (<\ ageOf(w)\ \ minConsentAgeOf(s)) \wedge$
$(hasHolderOfPr\ \ h\ w) \wedge (GiveConsentTo\ h\ e_p)\ ],$
$(isLawful\ e_p)\ ) \in C$

3 COMPLIANCE CHECKING IN RDFS/OWL

RDFs/OWL is nowadays *the* W3C standard language for the Semantic Web [1]. RDFs/OWL represents knowledge via *flat* sets of triples "`(subject, predicate, object)`", in which the `predicate` is an `rdf:Property` while the `subject` and the `object` can be any `rdfs:Resource`, including other `rdf:Property`(s). In other words, RDFs/OWL allows to treat `rdf:Property`(s) as first-order terms on which separately asserting other (meta-)properties.

It is then evident that reification is, in essence, the very same mechanism used to represent knowledge in RDFs/OWL, thus the idea of implementing reified I/O logic in the W3C standard.

Some proposals have been done to implement compliance checking in RDFs/OWL, e.g., [9] and [6]. In these approaches compliance checking is achieved by enriching the ontology with classes referring to sets of individuals compliant with the norms and by enforcing "is-a" inferences on these classes.

For instance, the OWL ontology used in [9] includes a class `Supplier` including individuals that supply consumers with some goods. Since suppliers are obliged to communicate their contractual conditions to their consumers (rule R1), the corresponding class includes a boolean datatype property `hasCommunicatedConditions` which is true for those suppliers that has complied with their obligation and false otherwise. The ontology includes then a class `SupplierR1compliant` defined as to include only individuals in `Supplier` for which `hasCommunicatedConditions` is true. Compliance checking is enforced by applying simple "is-a" inferences.

In the same spirit, [6] encodes in a fragment of OWL2 selected norms from Artt. 6, 7, 15, 23, and 30 of the GDPR, which concern data usage policies. Compliance on these policies is again implemented via "is-a" inferences.

While [9] and [6] are of course important contributions towards the same direction of research advocated here, it is not clear how to model exceptions in those frameworks. Furthermore, adding explicit classes specifically devoted to "collect" the individuals compliant with the norms, as well as introducing new ones to properly handle exceptions, does not appear to be an easy and intuitive solution.

The rest of the paper proposes to use SHACL as an alternative of the accounts in [9] and [6].

4 COMPLIANCE CHECKING IN SHACL

This paper proposes and makes initial investigations to encode legal rules in a formal language *different* from RDFs/OWL. This formal language is SHACL [2], proposed by W3C precisely for validation and inferences on RDFs/OWL graphs. The use of SHACL is currently a matter of ongoing research in the Semantic Web community (see [7], [24], among others).

SHACL appears to be the right formal language for modelling compliance checking, although so far it has been scarcely investigated to this end, preliminary works being [20], [21], and [8].

SHACL was originally proposed to define special conditions on RDFs/OWL graphs, called "SHACL shapes", more expressive than standard OWL cardinality and quantifier restrictions. RDFs/OWL graphs can be then *validated* against a set of such SHACL shapes.

However, SHACL "*may be used for a variety of purposes beside validation, including user interface building, code generation and data integration*" (cit. [2]). This paper adds a new use cases for SHACL in that it proposes to use it for serializing reified I/O logic formulae fit to check compliance.

In order to enhance the expressivity and the flexibility of the standard, a current W3C Working Group Note proposes to enrich SHACL shapes with *advanced features*[1] such as "SHACL rules" to derive inferred triples from asserted ones, prior to validation.

As explained in [25], SHACL rules can trigger ontological or non-ontological inferences. Ontological inferences derive facts that can be added to the model. On the other hand, non-ontological inferences have the sole purpose of aggregating data, without necessarily asserting them in the model, in order to facilitate validation.

5 SERIALIZING REIFIED I/O LOGIC IN SHACL

This paper represents the first attempt to investigate how to serialize reified I/O formula modeling obligations as SHACL shapes and reified I/O formula modeling constitutive rules as SHACL rules.

(6) shows the SHACL shape that serializes (2) above. Both require every personal data processing to be lawful.

(6)
```
CheckLawfulness
      rdf:type sh:NodeShape;
      sh:targetClass PersonalDataProcessing;
      sh:property [ sh:path is-lawful;
                    sh:hasValue "true"^^xsd:boolean; ];
```

In (6), "`sh:`" is SHACL namespace prefix. (6) is a `sh:NodeShape` requiring each individual of the `sh:targetClass` to satisfy the `sh:property`. The latter constrains the individuals reached from the `sh:targetClass` through the `sh:path` to satisfy `sh:hasValue`.

On the other hand, `PersonalDataProcessing`, `is-lawful`, and all other RDFs/OWL resources used in this paper are associated 1:1 with the predicates used in the reified I/O logic formulae such as (2), in the same way as the predicates occurring in the D-KB [28] are associated with RDFs/OWL resources from the PrOnto ontology [19], an OWL ontology proposed to conceptualize the data protection domain. Space constraints avoid to provide further details about the 1:1 mapping between reified I/O logic predicates and RDFs/OWL resources.

SHACL shapes refer to *constraints*, a solution that appears to be more intuitive and economical than overpopulating the ontology with extra classes as suggested in [9] and [6].

The validation facts, as well as new individuals, derived through SHACL are *not* mandatorily inserted in the ontology. The SHACL rules to model the reified I/O logic formulae in (3), (4), and (5) represent *non-ontological inferences*, in the sense explained in [25]: these rules are only *functional* to infer the truth value of `is-lawful` before the SHACL shape in (6) is validated.

(3), (4), and (5) are serialized in the SHACL rules in (7), (8), (9), and, below, (10).

(7)
```
sh:rule [rdf:type sh:TripleRule; sh:order 0;
      sh:subject sh:this;
      sh:predicate has-min-consent-age;
      sh:object [sh:path
        (has-theme has-personal-data
        is-personal-data-of has-member-state
        has-min-consent-age);]; ];
```

[1] See https://www.w3.org/TR/shacl-af

```
(8) sh:rule [rdf:type sh:TripleRule; sh:order 1;
      sh:condition [
        sh:property [sh:path has-min-consent-age;
          sh:minCount 1;];
        sh:property [
          sh:path (has-agent has-age);
          sh:lessThan has-min-consent-age;]; ];
      sh:subject [sh:path has-theme;];
      sh:predicate rdf:type;
      sh:object exceptionAgeDS; ];

(9) sh:rule [rdf:type sh:TripleRule; sh:order 2;
      sh:condition [
        sh:not [sh:property [sh:path has-theme;
          sh:class exceptionAgeDS;]; ]; ];
      sh:subject [sh:path has-theme;];
      sh:predicate is-lawful;
      sh:object "true"^^xsd:boolean; ];
```

The sh:targetClass of all these SHACL rules is GiveConsent. Rules are executed according to the sh:order, from the lowest to the highest value. Each rule in (7)-(9) makes a new assertion: the rdf:Property specified in the sh:predicate of the rule is asserted between the two RDFs/OWL resources in the sh:subject and the sh:object. The sh:subject and the sh:object may be the sh:targetClass itself (keyword "sh:this"), a resource reachable from the sh:targetClass through a path specified in sh:path, any other resource in the ontology, or a literal.

(7) is executed as first because its sh:order is "0". This rule sets the value of the property has-min-consent-age for each individual in the class GiveConsent. This values is set to the integer value reachable from the sh:path defined on sh:object in (7)). Specifically, this is the minimal consent age (has-min-consent-age) of the Member State (has-member-state) of the data subject owning the personal data (has-personal-data is-personal-data-of) involved in the personal data processing occurring as the theme of the GiveConsent instances (has-theme).

It is important to understand that has-min-consent-age will not be asserted on the individuals of GiveConsent also in the reference ontology, but only in the derived one. In other words, (7) is a non-ontological inference rule that collects/aggregates this value in GiveConsent for validation purposes only. After the validation, these values will be discharged.

Rule (8) compares the minimal consent age of the agents' Member State, just asserted by (7) on GiveConsent's instances, with the agents' age. The two rules are then executed in a pipeline, thanks to SHACL command sh:order. Mirroring these inferences in native RDFs/OWL seems to be more difficult in that the formalism does not allow to specify a priority between the inference rules.

When the agents' age has been specified (sh:minCount 1) and it is lower than (sh:lessThan) the minimal consent age of the Member State previously asserted by (7), rule (8) asserts the individual of PersonalDataProcessing in the has-theme property of the individual of GiveConsent as member of the class exceptionAgeDS (see rdf:type in sh:predicate).

Finally, (9) sets as true the property is-lawful of the instances of PersonalDataProcessing that do not (sh:not) belong to class exceptionAgeDS. (9) implements the reified I/O logic formula shown above in (3) and the SHACL operator sh:not the negation-as-failure (predicate *naf*) occurring therein. sh:not is in fact true when the ontology does not include any specific assertion of the personal data processing as member of the class exceptionAgeDS. In other words, since the close world assumption hold for both RDFs/OWL and SHACL, sh:not is true when it is either false or unknown whether the personal data processing belongs to this class.

Finally, (10) implements the reified I/O logic formula (5) above:

```
(10) sh:rule [rdf:type sh:TripleRule; sh:order 2;
       sh:condition [
         sh:property [sh:path
           (has-theme; has-personal-data
             is-personal-data-of has-age);
           sh:lessThan has-min-consent-age; ];
         sh:property [sh:path (has-theme
                               has-personal-data
                               is-personal-data-of
                               has-holder-of-pr);
           sh:equals has-agent;]; ];
       sh:subject [sh:path has-theme;];
       sh:predicate is-lawful;
       sh:object "true"^^xsd:boolean; ];
```

If the age of the data subject (has-age) who owns the personal data of the processing (has-personal-data is-personal-data) that is the theme of a GiveConsent individual (has-theme) is lower than (sh:lessThan) the minimal consent age of his/her Member State *and* the agent of this GiveConsent individual is the holder of the data subject's parental responsibility (has-holder-of-pr), then the boolean is-lawful is again set to true.

6 CONCLUSIONS

Reified I/O logic is a recent deontic logical framework explicitly designed to handle natural language semantics, i.e., to represent norms occurring in existing legislation such as the GDPR.

So far, the research in reified I/O logic has focused only on knowledge representation issues, specifically on how to use the formalism for representing contextual meaning of norms [3].

On the other hand, this paper is the first attempt to investigate computational issues in reified I/O logic, specifically how to represent the reified I/O logic if-then rules in a computable machine-readable format fit to enforce compliance checking.

This paper proposed to model regulative rules as SHACL shapes and constitutive rules as SHACL rules. SHACL shapes and rules are applied to RDFs/OWL models that describe states of affairs.

The solution proposed here is alternative to some recent approaches that model compliance checking on RDFs/OWL ontologies, e.g., [9] and [6].

On the other hand, the present work only represents the first step of a research endevour aiming at developing a full inference engine for reified I/O logic that implements and integrates all components involved in normative reasoning. Much further work needs to be done in order to obtain a formally well-defined framework, tested on existing industrial use cases.

Further directions of research include the automatic or semi-automatic generation of RDFs/OWL or SHACL assertions from legal texts, possibly via NLP (cf. [4], [5], [18]).

ACKNOWLEDGMENTS

This research has been supported by the Legal Innovation Lab Wales operation within Swansea University's Hillary Rodham Clinton School of Law. The operation has been part-funded by the European Regional Development Fund through the Welsh Government.

REFERENCES

[1] 2012. *Web Ontology Language (OWL).* Technical Report. W3C. https://www.w3.org/OWL

[2] 2017. *Shapes constraint language (SHACL).* Technical Report. W3C. https://www.w3.org/TR/shacl

[3] Cesare Bartolini, Andra Giurgiu, Gabriele Lenzini, and Livio Robaldo. 2016. Towards Legal Compliance by Correlating Standards and Laws with a Semi-automated Methodology. In *BNCAI (Communications in Computer and Information Science, Vol. 765).* Springer, 47–62.

[4] G. Boella, 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.

[5] Guido Boella, Luigi Di Caro, Daniele Rispoli, and Livio Robaldo. 2013. A System for Classifying Multi-label Text into EuroVoc. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Law* (Rome, Italy) *(ICAIL '13).* ACM, New York, NY, USA, 239–240.

[6] Piero A. Bonatti, Luca Ioffredo, Iliana M. Petrova, Luigi Sauro, and Ida Sri Rejeki Siahaan. 2020. Real-time reasoning in OWL2 for GDPR compliance. *Artificial Intelligence* 289 (2020).

[7] Julien Corman, Juan L. Reutter, and Ognjen Savkovic. 2018. Semantics and Validation of Recursive SHACL. In *The Semantic Web - ISWC 2018 - 17th International Semantic Web Conference, Monterey, CA, USA, October 8-12, 2018, Proceedings, Part I (Lecture Notes in Computer Science, Vol. 11136)*, Denny Vrandecic, Kalina Bontcheva, Mari Carmen Suárez-Figueroa, Valentina Presutti, Irene Celino, Marta Sabou, Lucie-Aimée Kaffee, and Elena Simperl (Eds.). Springer, 318–336.

[8] Christophe Debruyne, Harshvardhan J. Pandit, Dave Lewis, and Declan O'Sullivan. 2019. Towards Generating Policy-Compliant Datasets. In *13th IEEE International Conference on Semantic Computing, ICSC 2019, Newport Beach, CA, USA, January 30 - February 1, 2019.* IEEE, 199–203.

[9] Enrico Francesconi and Guido Governatori. 2019. Legal Compliance in a Linked Open Data Framework. In *Legal Knowledge and Information Systems - JURIX 2019: The Thirty-second Annual Conference, Madrid, Spain, December 11-13, 2019 (Frontiers in Artificial Intelligence and Applications, Vol. 322)*, Michal Araszkiewicz and Víctor Rodríguez-Doncel (Eds.). IOS Press, 175–180.

[10] G. Governatori, F. Olivieri, A. Rotolo, and S. Scannapieco. 2013. Computing Strong and Weak Permissions in Defeasible Logic. *Journal of Philosophical Logic* 6, 42 (2013), 799–829.

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

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

[13] J. R. Hobbs. 2008. Deep Lexical Semantics. In *Proc. of the 9th International Conference on Intelligent Text Processing and Computational Linguistics (CICLing-2008).* Haifa, Israel.

[14] R Kowalski and M Sergot. 1986. A Logic-based Calculus of Events. *New Generation Computing* 4, 1 (1986), 67–95.

[15] Tomer Libal and Alexander Steen. 2020. Towards an Executable Methodology for the Formalization of Legal Texts. In *Logic and Argumentation - Third International Conference, CLAR 2020, Hangzhou, China, April 6-9, 2020, Proceedings (Lecture Notes in Computer Science, Vol. 12061)*, Mehdi Dastani, Huimin Dong, and Leon van der Torre (Eds.). Springer, 151–165.

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

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

[18] Rohan Nanda, Luigi Di Caro, Guido Boella, Hristo Konstantinov, Tenyo Tyankov, Daniel Traykov, Hristo Hristov, Francesco Costamagna, Llio Humphreys, Livio Robaldo, and Michele Romano. 2017. A Unifying Similarity Measure for Automated Identification of National Implementations of European Union Directives. In *Proceedings of the 16th Edition of the International Conference on Articial Intelligence and Law (ICAIL 2017).* Association for Computing Machinery.

[19] Monica Palmirani, Michele Martoni, Arianna Rossi, Cesare Bartolini, and Livio Robaldo. 2018. PrOnto: Privacy Ontology for Legal Compliance. In *Proceedings of the 18th European Conference on Digital Government (ECEG).*

[20] Harshvardhan Jitendra Pandit, Declan O'Sullivan, and Dave Lewis. 2018. Exploring GDPR Compliance Over Provenance Graphs Using SHACL. In *Proc. of the Posters and Demos Track of the 14th International Conference on Semantic Systems co-located with the 14th International Conference on Semantic Systems (SEMANTiCS 2018), Vienna, Austria, September 10-13, 2018 (CEUR Workshop Proceedings, Vol. 2198)*, Ali Khalili and Maria Koutraki (Eds.).

[21] Harshvardhan J. Pandit, Declan O'Sullivan, and Dave Lewis. 2019. Test-Driven Approach Towards GDPR Compliance. In *Semantic Systems. The Power of AI and Knowledge Graphs*, Maribel Acosta, Philippe Cudré-Mauroux, Maria Maleshkova, Tassilo Pellegrini, Harald Sack, and York Sure-Vetter (Eds.). Springer International Publishing, 19–33.

[22] Xavier Parent 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.*

[23] Xavier Parent and Leendert van der Torre. 2014. "Sing and Dance!". In *Deontic Logic and Normative Systems*, Fabrizio Cariani, Davide Grossi, Joke Meheus, and Xavier Parent (Eds.). Springer International Publishing, 149–165.

[24] Paolo Pareti, George Konstantinidis, Fabio Mogavero, and Timothy J. Norman. 2020. SHACL Satisfiability and Containment. In *The Semantic Web - ISWC 2020 - 19th International Semantic Web Conference, Athens, Greece, November 2-6, 2020, Proceedings, Part I (Lecture Notes in Computer Science, Vol. 12506)*, Jeff Z. Pan, Valentina A. M. Tamma, Claudia d'Amato, Krzysztof Janowicz, Bo Fu, Axel Polleres, Oshani Seneviratne, and Lalana Kagal (Eds.). Springer, 474–493.

[25] Paolo Pareti, George Konstantinidis, Timothy J. Norman, and Murat Sensoy. 2019. SHACL Constraints with Inference Rules. In *The Semantic Web - ISWC 2019 - 18th International Semantic Web Conference, Auckland, New Zealand, October 26-30, 2019, Proceedings, Part I (Lecture Notes in Computer Science, Vol. 11778)*, Chiara Ghidini, Olaf Hartig, Maria Maleshkova, Vojtech Svátek, Isabel F. Cruz, Aidan Hogan, Jie Song, Maxime Lefrançois, and Fabien Gandon (Eds.). Springer, 539–557.

[26] L. Robaldo. 2010. Independent Set readings and Generalized Quantifiers. *The Journal of Philosophical Logic* 39(1) (2010), 23–58.

[27] L. Robaldo. 2011. Distributivity, Collectivity, and Cumulativity in terms of (In)dependence and Maximality. *The Journal of Logic, Language, and Information* 20(2) (2011), 233–271.

[28] L. Robaldo, C. Bartolini, M. Palmirani, A. Rossi, M. Martoni, and G. Lenzini. 2020. Formalizing GDPR provisions in reified I/O logic: the DAPRECO knowledge base. *The Journal of Logic, Language, and Information* 29 (2020). Issue 4.

[29] L. Robaldo 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 (2017). Issue 8.

[30] X. Sun and L. Robaldo. 2017. On the complexity of Input/Output logic. *The Journal of Applied Logic* 25 (2017), 69–88.