TLP **19** (1): 1–26, 2019. © Cambridge University Press 2018
doi:10.1017/S1471068418000339 First published online 14 September 2018

*Enabling Reasoning with LegalRuleML**

HO-PUN LAM
Data61, CSIRO, Sydney, NSW, Australia
(*e-mail:* brian.lam@data61.csiro.au)

MUSTAFA HASHMI
Data61, CSIRO, Brisbane, QLD, Australia
(*e-mail:* mustafa.hashmi@data61.csiro.au)

submitted 12 February 2017; accepted 28 May 2018

Abstract

In order to automate verification process, regulatory rules written in natural language need to be translated into a format that machines can understand. However, none of the existing formalisms can fully represent the elements that appear in legal norms. For instance, most of these formalisms do not provide features to capture the behavior of deontic effects, which is an important aspect in automated compliance checking. This paper presents an approach for transforming legal norms represented using legalruleml to a variant of modal defeasible logic (and vice versa) such that a legal statement represented using LegalRuleML can be transformed into a machine-readable format that can be understood and reasoned about depending upon the client's preferences.

KEYWORDS: Deontic logic, modal defeasible logic, legal reasoning, LegalRuleML, business contracts

1 Introduction

Generally, regulatory rules written in natural languages are required to transform into machine understandable formalisms before automated verification can take place. Over the years, numerous languages/standards, such as Rule Markup Language (RuleML) (RuleML Inc. 2000), Legal Knowledge Interchange Format (ESTRELLA Project 2008), Semantics of Business Vocabulary and Rules (SBVR) (OMG 2008b), PENELOPE (Goedertier and Vanthienen 2006), ConDec language (Pesic and Aalst 2006), ContractLog (Paschke *et al.* 2005), and OWL for Services[1] (Martin *et al.* 2004), have been proposed to facilitate this process. Each of these languages offer useful functionalities but is not free from shortcomings of (Gordon *et al.* 2009). For instance, ruleml is an XML-based standard language that enables users to use

* A preliminary version of this paper appeared in the *Proceedings of the 10th International Web Rule Symposium (RuleML 2016)*, J. J. Alferes, L. Bertossi, G. Governatori, P. Fodor and J. Hall, Eds, Springer, 2016, pp. 241–257.

[1] OWL for Services (OWL-S), originally called DAML-S, is an OWL-based ontology framework which provides a core set of construct for describing the properties and capabilities of web services in an unambiguous and machine interpretable way. http://www.daml.org/services/owl-s/

https://doi.org/10.1017/S1471068418000339 Published online by Cambridge University Press

different types of rules (such as derivation rules, facts, queries, integrity constraints, etc.) to represent different kinds of elements according to their needs. However, it lacks support for the use of deontic concepts, such as obligations, permissions, and prohibitions, making it impossible to handle cases with contrary-to-duty obligations (or reparational obligations) (Carmo and Jones 2002), which is not uncommon in legal contracts.

Grosof (2004) proposed to adopt courteous logic programming as the underlying execution model of RuleML rule base (for translating the clauses of a contract). A rather similar work was the Contract Tracking XML language developed by Farrell *et al.* (2004), with a computational model based on event calculus. However, these studies all suffer from the same problem as they do not consider normative effects.

Later, Governatori (2005) addressed the shortcomings of Grosof's work and extended Defeasible Logic (DL) (Nute 2001) with standard deontic operators for representing normative effects as well as semantic operator to deal with the contrary-to-duty obligations. This extended language also provides RuleML compliant data schemas for representing deontic elements and provides constructs to resolve some of the shortcomings that have been discussed in Gordon *et al.* (2009).

Following this line of research, in this paper, we focus on transforming the legal norms represented using LegalRuleML (OASIS LegalRuleML TC 2013), a normative extension of RuleML, into a variant of modal defeasible logic (MDL) (Governatori and Rotolo 2008b). This is due to the fact that legal statements are usually described in the form of natural language expressions, which cannot be applied to information system to automatically process it further. LegalRuleML, in this sense, provides a means for the legislators, legal practitioners, and business managers to formalize their legal documents into a machine-readable format such that information in the documents can be integrated, contrasted, and reused, but direct reasoning with the normative rules in the documents is still not possible. Hence, our work reported here makes it possible to use an implementation of Modal Defeasible Logic (MDL) as the engine to compute the extensions on the legal norms represented using LegalRuleML and reason on them.

The remainder of the paper is structured as follows: In Section 2, we tersely discuss a sample contract which will be used to motivate and illustrate how to use the logical framework and LegalRuleML in this context. Then, in Section 3, we will outline the logical framework required to represent contracts and its normative effects. Section 4 discusses core elements of a LegalRuleML document. The procedures to transform a legal theory represented using LegalRuleML to DL is discussed in Section 5. Related work is discussed in Section 6 followed by some concluding remarks and pointers for future work.

2 A sample contract

A *contract* is a set of declarative statements jointly agreed and performed by all parties that are involved in a particular task. It is a branch of the law of obligations which concerns about the rights and duties that arise from the agreed statements.

This paper is based on the analysis of the following sample "Contract of Services," adapted from Governatori (2005).

https://doi.org/10.1017/S1471068418000339 Published online by Cambridge University Press

Contract of Services

This deed of agreement is entered into effects between ABC company (to be known as Purchaser) and ISP plus (to be known as Supplier), whereas Purchaser desires to enter into an agreement to purchase from Supplier the application server (to be known as Goods) in this agreement. Both the parties shall enter into an agreement subject to the following terms and conditions:

(1) **Definitions and interpretations**
  (1.1) All prices are in Australian currency unless otherwise explicitly stated.
  (1.2) This agreement is governed by the Australian law and both the parties hereby agree to submit to the jurisdictions of the Courts of the Queensland with respect to this agreement.

(2) **Commencement and completion**
  (2.1) The contract enters into effects as Jan. 30, 2016.
  (2.2) The completion date is scheduled as Jan. 30, 2017.

(3) **Policy on price**
  (3.1) A *premium customer* is a customer who has spent more than $10,000 in goods. Premium customers are entitled a 5% discount on new orders.
  (3.2) Goods marked as *special order* are subject to a 5% surcharge. Premium customers are exempt from special order surcharge.
  (3.3) The 5% discount for premium customers does not apply for goods in promotions.

(4) **Purchase order**
  (4.1) The Purchaser shall follow the Supplier price lists on the supplier's website.
  (4.2) The Purchaser shall present Supplier with a purchase order for the provision of Goods within 7 days of the commencement date.

(5) **Service delivery**
  (5.1) The Supplier shall on receipt of a purchase order for Goods make them available within one working day.
  (5.2) Goods that are damaged during delivery shall be replaced by the Supplier within three working days from the notification by the Purchaser. Otherwise, the Supplier shall refund the Purchaser and pay the Purchaser a penalty of $1,000.
  (5.3) If for any reason the conditions stated in clauses 4 or 4 are not met, the Purchaser is entitled to charge the Supplier at the rate of $100 for per hour the Goods are not delivered.

(6) **Payments**
  (6.1) The payment terms shall be in full upon the receipt of invoice. An interest shall be charged at 5% on accounts not paid within 7 days of the invoice date. Another 1.5% interest shall be applicable if not paid within the next 15 days. The prices shall be as stated in the sales order unless otherwise agreed in writing by the Supplier.
  (6.2) Payments are to be sent electronically, and are to be performed under standards and guidelines outlined in PayPal.

(7) **Disputes** ...
(8) **Termination** ...

The agreement[2] covers a range of rule objectives such as roles of the involved parties (e.g., Supplier, Purchaser), authority, and jurisdiction (Australia, Queensland

[2] The contents of clauses 7 and 8 of the agreement have been omitted here as they are not relevant to the scope of this paper.

https://doi.org/10.1017/S1471068418000339 Published online by Cambridge University Press

Courts), deontic conditions associated with roles (permissions, prohibition), and temporal properties to perform required actions. A contract can be viewed as a legal document containing a finite set of articles (where each article contains a set of clauses and subclauses). The above-discussed agreement includes two main types of clauses namely (i) *constitutive clauses*, which define the basic concepts contained in this agreement; and (ii) *normative/prescriptive clauses*, which regulate the actions of Purchaser and Supplier for the performance of contract, and include deontic notions, e.g., obligations, permissions, etc.

3 The logical framework

DL (Nute 2001) is a rule-based skeptical approach to non-monotonic reasoning. It is based on a logic programming-like language and is a simple, efficient (Maher 2001), but flexible formalism capable of dealing with many intuitions of non-monotonic reasoning in a natural and meaningful way (Antoniou 2004). In this section, we sketch the basics of the logical apparatus used in the paper. Basically, we will combine three logical components, namely (i) DL, (ii) deontic concepts, and (ii) a fragment of logic related to normative violations, such as contrary-to-duty obligations.

The primary use of DL in the present context is aimed at facilitating the representation of different types of statements in legalruleml into different types of rules according to their nature, and to resolve the conflicts that may arise from the clauses of a contract using priorities and override predicates.

3.1 Defeasible logic

A *defeasible theory* (Antoniou *et al.* 2001) D as a triple $(F, R, >)$, where (i) F is a set of *facts* or *indisputable* statements, (ii) R is the set of rules, and (iii) $>$ is an acyclic *superiority relation* on R. Given a set PROP of *propositional atoms*, the set $\mathsf{Lit} = \mathsf{PROP} \cup \{\neg p \,|\, p \in \mathsf{PROP}\}$ denotes the set of *literals*. If q is a literal, then $\sim q$ denotes its complement; if q is a positive literal p, then $\sim q$ is $\neg p$, and if q is $\neg p$, then $\sim q$ is p.

Hence, given Lbl a set of arbitrary labels, every rule in R is of the form:

$$r : A(r) \hookrightarrow C(r)$$

where

- $r \in \mathsf{Lbl}$ is the unique identifier of the rule;
- $A(r) = \phi_1, \ldots, \phi_n$, the *antecedent* of the rule, is a finite set of literals denoting the premises of the rule, and can be omitted if it is *empty*;
- $\hookrightarrow \in \{\rightarrow, \Rightarrow, \leadsto\}$ denotes the *type* of the rule;
- $C(r)$ is the *consequent* (or *head*) of the rule, contains a single literal.

The intuition behind different arrows is the following. DL support three types of rules namely, *strict rules* $(r : A(r) \rightarrow C(r))$, *defeasible rules* $(r : A(r) \Rightarrow C(r))$, and *defeaters* $(r : A(r) \leadsto C(r))$. Strict rules, in the classical sense, are the rules that the

https://doi.org/10.1017/S1471068418000339 Published online by Cambridge University Press

conclusion follows every time the antecedents hold; a defeasible rule is allowed to assert its conclusions in case there is no contrary evidence to it. Finally, defeaters suggest there is a connection between its premises and its conclusion(s) but not strong enough to warrant the conclusion on its own; they are used to defeat rules for the opposite conclusion(s).

DL is a *skeptical* non-monotonic formalism meaning that it does not support contradictory conclusions. Instead, it seeks to resolve conflicts. In case there is some support for concluding A but there is also support for concluding $\neg A$, DL does not conclude either of them. However, if the support for A is stronger than the support of $\neg A$, then A is concluded. Here, the superiority relation $>$ is used to describe the relative strength of rules on R. When $r_1 > r_2$, then r_1 is called *superior* to r_2, and r_2 *inferior* to r_1. Intuitively, $r_1 > r_2$ expresses that r_1 overrides r_2 if both rules are applicable[3].

DL differentiates positive conclusions from negative conclusions, that is, literals that can be proved or literals that are refuted. In addition, it is able to determine the strength of conclusions, i.e., whether something is concluded using only strict rules and facts, or whether we have a defeasible conclusion—a conclusion that can be retracted if more evidence is provided. Accordingly, for a literal q, we have the following four types of conclusions, called *tagged* literals:

- $+\Delta q$ meaning that q is definitely provable in D (i.e., using only facts or strict rules).
- $-\Delta q$ meaning that q is definitely rejected in D.
- $+\partial q$ meaning that q is defeasibly provable in D.
- $-\partial q$ meaning that q is defeasibly rejected in D.

Provability is based on the concept of *derivation* (or *proof*) in D satisfying the proof conditions. Informally, strict derivations are obtained by forward chaining of strict rules while a defeasible conclusion q can be derived if there is a rule whose conclusion is q, and its (prerequisite) antecedent has either already been proved or given in the case at hand (i.e., facts), and any stronger rules whose conclusion is $\neg p$ has prerequisite that it failed to be derived. In other words, a conclusion q is defeasibly derivable when (i) q is a fact; or (ii) there is an applicable strict or defeasible rule for q, and either all rules for $\neg p$ are discarded (i.e., inapplicable) or every rule for $\neg p$ is weaker than an applicable rule for q.

To illustrate the inferential mechanism of DL, let us assume we have a defeasible theory containing the following rules:

$$\begin{aligned}
r_1:&\quad SpecialOrder(X) \Rightarrow \neg Discount(X)\\
r_2:&\quad PremiumCustomer(X) \Rightarrow Discount(X)\\
r_3:&\quad Promotion(X) \Rightarrow \neg Discount(X)
\end{aligned}$$

[3] Here, the notion of a rule is *applicable* means that all the antecedents of the rule are provable; a rule is *discarded* if at least one of its antecedents is refuted; a rule is *defeated* if there is a (stronger) rule for the complement of the conclusion that is applicable.

https://doi.org/10.1017/S1471068418000339 Published online by Cambridge University Press

where $>= \{r_3 > r_2, r_2 > r_1\}$. The theory states that products in promotion are not discounted, and so are special orders except when the order is placed by a premium customer, who are normally entitled to a discount (see clause 3.1. of the contract).

In a scenario where a customer would like to buy a product with special order, then we can conclude that the price has to be calculated with *no* discount since rule r_2 is not applicable. In case where the order is received from a premium customer and the product is not in promotion, then the customer is entitled to receive a discount, as rule r_2 is now applicable and stronger than r_1, while r_3, which is stronger than r_2, is not applicable (i.e., the product is not in promotion).

The set of conclusions is finite and can be computed in linear time (Maher 2001). Besides, the reasoning engine can be implemented as a chip (Song 2008). Over the years, various efficient and powerful implementations have been developed (Maher *et al.* 2001; Bassiliades *et al.* 2004; Antoniou and Bikakis 2007; Lam and Governatori 2009) to facilitate the theoretical and applications development of DL. For a full presentation and proof conditions of DL, please refer to Antoniou *et al.* (2001).

Recently, some studies have attempted to relate DL with other logical formalisms through its argumentation semantics (Governatori *et al.* 2004). For instance, Lam *et al.* (2016) have compared the ambiguity propagation variant of DL (Antoniou *et al.* 2000) with ASPIC$^+$ (Prakken 2010; Modgil and Prakken 2013; Modgil and Prakken 2014) based on the acceptability of arguments, and proposed a mapping from ASPIC$^+$ to DL. Hecham *et al.* (2017), on the other hand, proposed a hypergraph-based algorithm for reasoning conclusions from existential rules in *Defeasible Datalog*$^\pm$ (Martinez *et al.* 2014; Deagustini *et al.* 2015)—an extension of Datalog$^\pm$ (Calì *et al.* 2012) which includes defeasible facts and defeasible rules, but allows weak negation instead of classical negation (as in DL) in the body of the rules. Their approach has overcome the non-deterministic issues that may appear during the reasoning process and has been implemented as a tool called DEFT (Defeasible Datalog$^\pm$ Tool).

3.2 Modal Defeasible Logic (MDL)

Having the basics of DL is not sufficient enough. The most essential part of developing a legal reasoning system is on creating the framework, norms, etc., for representing the normative behavior of a contract.

Here, we follow the line of work by Governatori and Rotolo (2008a), Governatori and Rotolo (2008b), and Lam and Governatori (2013), and extend DL with the support of modalities. Let MOD denotes the set of modal operators and the set of *modal literals* be $\mathsf{ModLit} = \{Xl, \neg Xl \mid l \in \mathsf{Lit}, X \in \mathsf{MOD}\}$[4].

To enhance the expressiveness of a rule to encode chains of obligations and violations, following the ideas of Governatori and Rotolo (2006), a sub-structural operator $\otimes$ is introduced to capture an obligation and the obligations arising in response to the violation of the obligation. Thus, given an expression like $a \otimes b$, the

[4] Notice that, here, we do not allow nesting of modal operators. This is a simplification aimed at keeping the system manageable, but does not pose severe limitations for our purpose.

https://doi.org/10.1017/S1471068418000339 Published online by Cambridge University Press

intuitive reading is that if a is possible, then a is the first choice and b is the second one; if $\neg a$ holds, i.e., a is violated, then b is the actual choice. That is, the $\otimes$-operator is used to build chains of preferences, called $\otimes$-*expression*, such that (i) each literal is an $\otimes$-expression; (ii) if A is an $\otimes$-expression and b is a (modal) literal, then $A \otimes b$ is an $\otimes$-expression, whose properties are given in Definition 1 below.

Definition 1 (*Lam and Governatori (2013)*)
A $\otimes$-expression is a binary operator satisfying the following properties:

(1) $a \otimes (b \otimes c) = (a \otimes b) \otimes c$ (associativity).
(2) $\bigotimes_{i=1}^{n} a_i = (\bigotimes_{i=1}^{k-1} a_i) \otimes (\bigotimes_{i=k+1}^{n} a_i)$ where exists j such that $a_j = a_k$ and $j < k$ (duplication and contraction).

And the definition of rule in MDL becomes the following.

Definition 2
A rule is an expression

$$r : A(r) \hookrightarrow_{\Box} C(r)$$

where

- $r \in \mathsf{Lbl}$ is the unique identifier of the rule;
- $A(r) = \phi_1, \ldots, \phi_n$ is a final set of (modal) literals denoting the premises of the rule, and can be omitted if it is *empty*;
- $\hookrightarrow \in \{\rightarrow, \Rightarrow, \leadsto\}$ denotes the *type* of the rule and $\Box$ is a modal operator;
- $C(r)$ is the *consequent* (or *head*) of the rule, which can be either a single (modal) literal, or an $\otimes$-expression if $\hookrightarrow = \Rightarrow$.

The derivation of conclusions in MDL is similar to that of DL but is beyond the scope of this paper. For a full description of the proof conditions and algorithms to compute the extensions of MDL, please refer to Governatori and Rotolo (2008b) and Lam and Governatori (2013) for details.

Throughout the paper, we use the following abbreviations on set of rules: R_s (R_d) denotes the set of strict (defeasible) rules, $R[q]$ denotes the set of rules with consequent q. For a rule $r \in R$, we use $C(r, i)$ denotes the ith (modal) literal that appears in $C(r)$, and $R[c_i = q]$ denotes the set of rules with head $\otimes_i^n c_i$ and $c_i = q$ for some $i \in \{1, n\}$.

4 LegalRuleML: the legal rule markup language

LegalRuleML (Palmirani *et al.* 2015) is a rule interchange language proposed by OASIS, which extends RuleML with features specific to the legal domain (Athan *et al.* 2015). It aims to bridge the gap between natural language descriptions and semantic norms (Athan *et al.* 2013), and can be used to model various laws, rules, and regulations by translating the compliance requirements into a machine-readable format (Hashmi *et al.* 2016).

A LegalRuleML document is structured into three main components namely, *metadata*, *context*, and *statements*, as depicted in Figure 1. The metadata component

https://doi.org/10.1017/S1471068418000339 Published online by Cambridge University Press

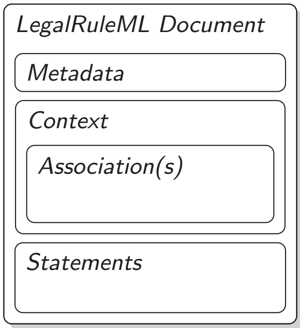

Fig. 1. LegalRuleML document structure.

contains the legal sources of the norms modeled by the document, the temporal information about the legal sources, and the document itself, jurisdiction where the norms are applicable, and the details concerning the authorities for the legal sources and the document. The context component, on the other hand, is used to store important relationships and corresponding information between metadata and the rules (or fragment of them). This is due to the fact that the same rule can be interpreted differently due to a variety of parameters such as jurisdiction and temporal parameters that can be changed over time. To cater for such situations, the description of all characteristics of a particular rule can be stored inside the context component and will be extracted according to the context as required. The statement component contains formal representation of the legal norms in the form of rule statements.

Notice that LegalRuleML supports modeling of defeasibility as within the law. The intended reading of defeasible rules in LegalRuleML is that the conclusion tentatively holds when the antecedent of the rule is supported by the evidence/facts of a case. Then, the conclusion will further be evaluated if contradictory conclusion(s) with valid arguments has appeared, or exceptions have been identified. In addition, it also provides features to model various effects that follow from applying rules, such as obligations, permissions, and prohibitions, and can specify preferences among them.

In the next section, we will provide the description of how different types of rules are represented using LegalRuleML and describe the way of how they can be transformed into the DL framework that we have described in the previous section.

5 Transforming LegalRuleML into DL

A contract written in LegalRuleML is not intended to be executed directly, but the business logic can be transformed into a target language of a rule-based system to execute. In this section, we are going to explore the building blocks of LegalRuleML and propose a method to transform legal norms represented in LegalRuleML into a DL theory. Since LegalRuleML is essentially an extension of RuleML, here we only highlight the differences and identify the additions to faithfully represent legal norms.

https://doi.org/10.1017/S1471068418000339 Published online by Cambridge University Press

5.1 Premises and conclusions

The first thing we have to consider is the representation of predicates (atoms) to be used in premises or conclusions in LegalRuleML. LegalRuleML extends the construct from RuleML and represents a predicate as an *n*-ary relation, and is defined using an element <ruleml:Atom>[5]. Normative effects of an atom, on the other hand, are captured by embedding the atom inside a deontic element. The legal concepts such as *obligation* (<lrml:Obligation>), *permission* (<lrml:Permission>), *prohibition* (<lrml:Prohibition>), and *right* (<lrml:Right>)[6] forms the basic deontic elements in LegalRuleML. Further refinements are possible by (i) providing an iri[7] attribute of a deontic specification, or (ii) using the <lrml:Association> and <lrml:toTag> elements to link a deontic specification to its meaning with the <lrml:appliesModality> element[8].

```
<lrml:Associations>
  <lrml:Association key="asc1">
    <lrml:appliesModality iri="ex:achievementObligation"/>
    <lrml:toTarget keyref="#oblig101"/>
  </lrml:Association>
</lrml:Associations>

<lrml:Obligation key="oblig101">
  <ruleml:Atom key=":atom109">
    <ruleml:Rel iri="pay"/>
    <ruleml:Ind>Purchaser</ruleml:Ind>
    <ruleml:Ind>receivedReciept</ruleml:Ind>
    <ruleml:Ind>Supplier</ruleml:Ind>
  </ruleml:Atom>
</lrml:Obligation>
```

Accordingly, the above listing represents a modal literal OBL*pay*(*purchaser*,*receivedReceipt*,*supplier*) for the clause 6 in the contract that is true when *purchaser* has the obligation[9] to pay the *supplier* upon payment receipt[10].

[5] Elements from LegalRuleML and elements inherited from RuleML will be prefixed with **lrml** and **ruleml**, respectively. Information about transforming norms represented using RuleML to DL can be found in Governatori (2005). The attributes **key** and **keyref** in LegalRuleML correspond to a unique identifier and reference to an element, respectively. Elements inside a LegalRuleML document can be referenced/link together using these attributes. However, since they are not specifically relevant to the discussion here and thus they are omitted.

[6] Note that the element *right* here is different from the element *right* in RuleML. In LegalRuleML, the element *right* is a deontic specification that gives a permission to a party and implies that there is no obligation or prohibition on the other parties (Palmirani *et al.* 2015), while the element *right* in RuleML means the right-hand side of a rule.

[7] An **iri** attribute on a node element in LegalRuleML corresponds to an <owl:sameAs> relationship in the abstract syntax.

[8] The <lrml:Association> element is used to store the metadata information that elements in a LegalRuleML theory can associate with, while the element <lrml:toTarget> is used to indicate which element(s) the association is going to be applied to.

[9] There are several types of obligations based on temporal validity and effects they produce, e.g., *achievement, maintenance,* etc., see Hashmi *et al.* (2016) for details.

[10] In this paper, we are going to use the modal operator OBL for obligation, PER for permission, FOR for prohibition (forbidden).

https://doi.org/10.1017/S1471068418000339 Published online by Cambridge University Press

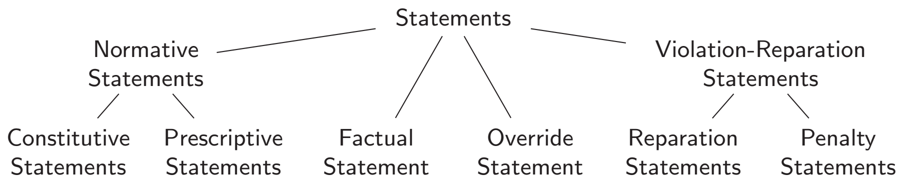

Fig. 2. Types of statements in LegalRuleML.

5.2 Rules and rulebases

Norms in LegalRuleML are represented as collections of statements, and can be classified into four different types according to their nature, namely, *norm statements*, *factual statements*, *override statements*, and *violation-reparation statements*. These can be further classified into subtypes, as depicted in Figure 2.
In this section, we are going to explore different types of statements and describe how they can be transformed into rules in DL. To facilitate our discussion, we have the following definition.

Definition 3 (*Compliance and violation (Palmirani et al. 2015)*)

- A *compliance* is an indication that an obligation has been fulfilled or a prohibition has not been violated.
- A *violation* is an indication that an obligation or prohibition has been violated.

5.2.1 Norm statements

Legal norms, in general, can be classified into *constitutive norms* (which is used to represent *institutional facts* (Searle 1997) and provide definitions of terms and concepts in a jurisdiction (Palmirani *et al.* 2015)) and *prescriptive norms* (which specify the deontic behavior and effects of a legal system). These can be represented as *constitutive statements* (`<lrml:ConstitutiveStatement>`) and *prescriptive statements* in LegalRuleML (`<lrml:PrescriptiveStatement>`), respectively, to allow new information to be derived using existing rules.

The following is an example of a prescriptive statement representing the first statement of the clause 3 of the service contract where *goods* marked with *special order* are subject to a surcharge.

```
<lrml:PrescriptiveStatement key="r1">
  <ruleml:Rule key=":ruletemplate1">
    <lrml:hasStrength>
      <lrml:DefeasibleStrength key="str1"
        iri="http://example.org/legalruleml/ontology#defeasible1"/>
    </lrml:hasStrength>
    <ruleml:if>
      <ruleml:And>
        <ruleml:Atom key=":atom2">
          <ruleml:Rel iri=":specialOrder"/>
          <ruleml:Var>X</ruleml:Var>
        </ruleml:Atom>
      </ruleml:And>
    </ruleml:if>
    <ruleml:then>
```

https://doi.org/10.1017/S1471068418000339 Published online by Cambridge University Press

```
      <lrml:Obligation>
        <ruleml:Atom key=":atom3">
          <ruleml:Rel iri=":surcharge"/>
          <ruleml:Var>X</ruleml:Var>
        </ruleml:Atom>
      </lrml:Obligation>
    </ruleml:then>
  </ruleml:Rule>
</lrml:PrescriptiveStatement>
```

Similar to the *derivation rules* in RuleML, every constitutive/prescriptive statement has two parts: *conditions* (`<ruleml:if>`), which specify the conditions (using a conjunction of formulas and may possibly empty), and *conclusion* (`<ruleml:then>`), the effects of the rule. Additionally, a separate element (`<lrml:hasStrength>`) can be used to specify the strength of the rule.

Both rules can have deontic formulas as their preconditions (body). However, the difference between the two statements is in the contents of the head, where the head of a prescriptive statement is a list of deontic formulas. In contrast, the head of a constitutive statement cannot be a deontic formula (Palmirani *et al.* 2015).

In this perspective, a constitutive/prescriptive statement can be transformed into a rule of the form

$$label : body \hookrightarrow head,$$

where *label* is the key of the statement, $\hookrightarrow \in \{\rightarrow, \Rightarrow, \leadsto\}$ is the rule type, *body* and *head* are the set of (modal) literals inside the `<ruleml:if>` and `<ruleml:then>` elements of the statement, respectively. Unless otherwise specified, due to its nature, the rule modeled using a constitutive statement will be transformed into a strict rule, while the rule modeled using prescriptive statement will be transformed into a defeasible rule. Thus, the statement above will be transformed to the defeasible rule below[11]:

$$r_1 : specialOrder(X) \Rightarrow \text{OBL } surcharge(X)$$

5.2.2 Factual statements

Factual statements, in essence, are the expression of facts and can be considered as a special case of norm statements without the specification of premises. They denote

[11] Note that in some variants of DL, new types of rules can be created for the deontic operator to differentiate between normative and definitional rules (Governatori and Rotolo 2004). For instance, the rule r_1 above will become $specialOrder \Rightarrow_{\text{OBL}} surcharge$ indicating a new type of rule relative to the modal operator OBL. However, we do not utilize this approach here as this will limit ourselves such that only one type of modality can appear in the head of the rule. As it is possible that different logics/semantics can be used to reason on the rules generated using the constitutive and prescriptive statements, using such approach will limit the logic that we can use when reasoning the rules. For example, given the skeptical nature of the reasoning, conclusions of conflicting literals are considered both as not provable and will ignore the reasons why they were when we use them as premises of further arguments, which is normally the case for rules generated using constitutive statements. However, in some legal settings, we may want this ambiguity to be propagated along the line of the inference, which is not uncommon for rules generated using prescriptive statements. In the first case, we speak of ambiguity blocking, in the latter case of ambiguity propagation (of DL). As the discussion of this is beyond the scope of this paper, reader interested in these topics may refer to Antoniou *et al.* (2000) and Lam and Governatori (2011) for details.

https://doi.org/10.1017/S1471068418000339 Published online by Cambridge University Press

a simple piece of information that is deemed to be true. Below is an example of a factual statement in LegalRuleML representing the fact $premiumCustomer(JohnDoe)$, meaning that "JohnDoe" is a premium customer.

```
<lrml:FactualStatement key="fact1">
  <lrml:hasTemplate>
    <ruleml:Atom key=":atom11">
      <ruleml:Rel iri=":premiumCustomer"/>
      <ruleml:Ind iri=":JohnDoe"/>
    </ruleml:Atom>
  </lrml:hasTemplate>
</lrml:FactualStatement>
```

5.2.3 Override statements

To handle defeasibility, LegalRuleML uses *override statements* (`<lrml:OverrideStatement>`) to capture the relative strength of rules that appear in the legal norms. The element `<lrml:Override>` defines the relationship of superiority such that the conclusion of $r2$ overrides the conclusion of $r1$ (where $r1$ and $r2$ are the keys of statements in the legal theory, as shown below) if both statements are applicable.

Consider again clause 3 of the contract where a *premium customer* is exempted from the surcharge for goods marked as *Special Orders*, which can be modeled as the rules below:

$$r_1:\ specialOrder(X) \Rightarrow \text{OBL}\ surcharge(X)$$
$$r_2:\ specialOrder(X), premiumCustomer(Y) \Rightarrow \text{OBL}\neg surcharge(X)$$

In the above example, the conclusion of r_2 takes the precedence over the conclusion of r_1, if the order was made from a *premium customer*. The following listing illustrates this using an `<lrml:OverrideStatement>` element.

```
<lrml:OverrideStatement>
  <lrml:Override over="#r2" under="#r1"/>
</lrml:OverrideStatement>
```

In DL terms, this construct defines a superiority relation between $r_2 > r_1$, where r_1 and r_2 are the rule labels of the rules generated using the statements $r1$ and $r2$ in the legal norms, respectively.

5.2.4 Violation-reparation statements

Obligations can be violated, meaning that the content of the obligation has not been achieved. However, a violation may not result in inconsistency or a termination of interaction as a penalty can be introduced to compensate the violation (Hashmi *et al.* 2016).

In LegalRuleML, a *violation-reparation statement* is the type of statement concerning what actions are required when an obligation is violated. It provides two basic building blocks to model this, namely, *penalty statements* (`<lrml:PenaltyStatement>`) and *reparation statements* (`<lrml:ReparationStatement>`), as shown below.

https://doi.org/10.1017/S1471068418000339 Published online by Cambridge University Press

```
<lrml:ReparationStatement key="reps1">
  <lrml:Reparation key="rep1">
    <lrml:appliesPenalty keyref="#pen1"/>
      <lrml:toPrescriptiveStatement
                keyref="#ps1"/>
  </lrml:Reparation>
</lrml:ReparationStatement>
```

```
<lrml:PenaltyStatement key="pen1"
    >
  <lrml:SuborderList>
    list of deontic formulas
  </lrml:SuborderList>
</lrml:PenaltyStatement>
```

Essentially, penalty statements model sanctions and/or correction for a violation of a specified rule as outlined in the reparation statement; reparation statements bind a penalty statement to the appropriate prescriptive statement and apply the penalty when a violation occurs. Elements in the `<lrml:SuborderList>` (inside the `<lrml:PenaltyStatement>`) is a list of deontic formulas, i.e., formula of the form $Op\,A$, where Op is a deontic operator and A is a literal, such that a formula in the list holds if all deontic formulas that precede it in the list have been violated (Palmirani *et al.* 2015).

To transform these statements into DL rules, we can utilize the $\otimes$-expression that we described in Section 3 by appending the list of modal literals that appear in the penalty statements at the end of original rule. As an example, consider the penalty statement (in clause 6.1 of the contract) for not paying invoice within the deadline, and assume that the two model literals OBL $payWith5\%Interest$ and OBL $payWith6.5\%Interest$ are transformed from the suborder list inside the penalty statement. Then the prescriptive statement $ps1$ will be updated from

$$ps1 : goods(X), invoice(X) \Rightarrow \text{OBL}\ payIn7days(X)$$

to

$$ps1 : goods(X), invoice(X) \Rightarrow \text{OBL}\ payIn7days(X) \otimes \text{OBL}\ payWith5\%Interest(X) \otimes \text{OBL}\ payWith6.5\%Interest$$

5.3 Other constructs

Up to this point, the transformations described have been relatively simple. However, the transformations cast a wider net than is relevant to the discussion here; thus, for our present purpose, we limit ourselves to two of the statement/rule-related elements introduced in LegalRuleML, which is not that intuitive.

In legal contract, there are normative effects, such as *obligations*, *permissions*, and *prohibitions*, that follow from applying rules. However, there are situations where rules are also used to regulate methods for detecting violations or to determine normative effects triggered by other norm violations, which are meant to compensate or repair violations (Palmirani *et al.* 2015). In this regard, LegalRuleML provides two deontic elements that can be used to *determine* whether an obligation or a prohibition of an object has been fulfilled (`<lrml:Compliance>`) or violated (`<lrml:Violation>`).

Consider the listing below which represents the rule:

$$ps2 : \text{PER}\ rel1(X), \text{OBL}\ rel2(X) \Rightarrow \text{FOR}\ \neg rel3(X).$$

https://doi.org/10.1017/S1471068418000339 Published online by Cambridge University Press

```
<lrml:PrescriptiveStatement key="ps2">
  <ruleml:Rule key=":ruletemplate2">
    <ruleml:if>
      <ruleml:And key=":and1">
        <lrml:Violation keyref="#ps3"/>
        <lrml:Permission>
          <ruleml:Atom key=":atom4">
            <ruleml:Rel iri=":rel1"/>
            <ruleml:Var>X</ruleml:Var>
          </ruleml:Atom>
        </lrml:Permission>
        <lrml:Obligation key="oblig1">
          <ruleml:Atom key=":atom5">
            <ruleml:Rel iri=":rel2"/>
            <ruleml:Var>X</ruleml:Var>
          </ruleml:Atom>
        </lrml:Obligation>
      </ruleml:And>
    </ruleml:if>
    <ruleml:then>
      <lrml:Prohibition key="prohib1">
        <ruleml:Neg key=":neg1">
          <ruleml:Atom key=":atom6">
            <ruleml:Rel iri=":rel3"/>
            <ruleml:Var>X</ruleml:Var>
          </ruleml:Atom>
        </ruleml:Neg>
      </lrml:Prohibition>
    </ruleml:then>
  </ruleml:Rule>
</lrml:PrescriptiveStatement>
```

However, here we have a violation element appearing in the body as a prerequisite to activate the rule, meaning that the referenced element (*ps*3 in this case) has to be violated or the rule *ps*2 cannot not be utilized. Accordingly, we have two cases: either (i) the referenced element is a modal literal, or (ii) the referenced element is a rule.

5.3.1 *Case 1: Referenced element is a literal*

The former is a simple case. If the referenced element is a literal, essentially it acts as a precondition to activate the rule. It is practically the same as appending the violation (respectively, compliance) condition to the body of the rule, as shown below:

$$ps2 : \mathrm{PER}\, \mathit{rel1}\,(X), \mathrm{OBL}\, \mathit{rel2}\,(X), \mathit{violate}(p) \Rightarrow \mathrm{FOR}\, \neg \mathit{rel3}\,(X).$$

where p is the referenced literal, $\mathit{violate}(p)$ (respectively, $\mathit{comply}(p)$) is a transformation, as defined in Table 1, that transforms the (modal) literal p into a set of literals that needs to be derived in order to satisfy the condition of violation (compliance). This is due to the fact that, basically, for a literal q, a situation is violated when we have $\mathrm{OBL}\, q$ and $\neg q$ (for obligation), or $\mathrm{FOR}\, q$ and q (for forbidden or prohibition), while a situation is compliance when we have $\mathrm{OBL}\, q$ and q (for obligation), or $\mathrm{FOR}\, q$ and $\neg q$ (for forbidden or prohibition). For instance, if *ps*3 is the modal literal $\mathrm{OBL}\, q$, then the rule *ps*2 above will be updated as follows:

$$ps2 : \mathrm{PER}\, \mathit{rel1}\,(X), \mathrm{OBL}\, \mathit{rel2}\,(X), \mathrm{OBL}\, q, \neg q \Rightarrow \mathrm{FOR}\, \neg \mathit{rel3}\,(X).$$

https://doi.org/10.1017/S1471068418000339 Published online by Cambridge University Press

Table 1. *Requirements to determine whether a literal is compliant or violated*

	q	OBL q	FOR q
Compliance	q	OBL q , q	FOR q , $\neg q$
Violation	$\neg q$	OBL q , $\neg q$	FOR q , q

However, the case is somewhat complex when the referenced deontic element appears at the head of the statement, as shown in the listing below:

```
1  <lrml:PrescriptiveStatement key="ps4">
2    <ruleml:Rule key=":ruletemplate3" keyref=":ruletemplate2">
3      <ruleml:if>
4        ⋮
5      <ruleml:if>
6      <ruleml:then>
7        <lrml:SuborderList>
8          <lrml:Obligation key="obl1">
9            <ruleml:Atom key=":atom26">
10             <ruleml:Rel iri=":rel3"/>
11             <ruleml:Var>X</ruleml:Var>
12           </ruleml:Atom>
13         </lrml:Obligation>
14         <ruleml:And>
15           <lrml:Violation keyref="#ps5"/>
16           <lrml:Obligation key="obl2">
17             <ruleml:Atom key=":atom27">
18               <ruleml:Rel iri=":rel4"/>
19               <ruleml:Var>X</ruleml:Var>
20             </ruleml:Atom>
21           </lrml:Obligation>
22         </ruleml:And>
23       </lrml:SuborderList>
24     </ruleml:then>
25   </ruleml:Rule>
26 </lrml:PrescriptiveStatement>
```

Here, OBL *rel4* (Lines 16–21) is derivable only when the modal literal OBL *rel3* (Lines 8–13) is defeated and the reference literal $ps5$ (Line 15) is violated, which can be considered as a precondition of making OBL *rel4* becomes applicable and can be represented as the rule below:

$$ps4 : A(ps4) \Rightarrow \text{OBL}\, rel3 \otimes (violate ps5 \Rightarrow \text{OBL}\, rel4)$$

where $A(ps4)$ is the antecedent of the rule $ps4$. Notice that, here we have abused the use of notations by nesting a sub-rule into the head of the rule. However, such nested structure is *not* supported semantically in DL. To resolve this issue, we have to modify the statement based on its expanded form.

Definition 4 ($\otimes$*-expansion*)

Let $D = (F, R, >)$ be a defeasible theory, and let Σ be the language of D. We define $reduct(D) = (F, R', >')$ where for every rule $r \in R_d$ with a $\otimes$-expression $c_1 \otimes \cdots \otimes c_n$,

https://doi.org/10.1017/S1471068418000339 Published online by Cambridge University Press

appears in its head:

$$R' = R \setminus R_d \cup \{\ r\ : A(r) \Rightarrow c_1 \\ r'\ : A(r), violate(c_1) \Rightarrow c_2 \otimes \cdots \otimes c_n\}$$
$$\forall r', s' \in R', r' > s' \Leftrightarrow r, s \in R \text{ s.t. } r' \in reduct(r), s' \in reduct(s), r > s.$$

Definition 5
Let $D = (F, R, >)$ be a defeasible theory, Σ be the language of D, and $\forall r \in R_d$, $C(r) = p$ is a (modal) literal. We define $\mathscr{T}(D) = (F, R', >)$ where $\forall r \in R_d$:

$$R' = R \setminus R_d \cup \{r\ : A(r), verify(p) \Rightarrow p\}$$

where $verify(p)$ is defined as

$$\begin{cases} violate(e) & \text{if a violation element is attached to the element } p, \\ comply(e) & \text{if a compliance element is attached to the element } p, \\ \emptyset & \text{otherwise.} \end{cases}$$

where e is the literal referenced by the deontic element attributed to C.

Here, we can first exclude the deontic elements in the rule head and generate the rule based on $\otimes$-expression. Then, we can apply Definition 4 recursively to transform the generated rule into a set of rules with single (modal) literal in its head. Afterwards, similar to the case discussed before, we can append the deontic element to the body of the rule(s) (Definition 5), as an inference condition, where appropriate. Hence, the statement $ps4$ above can be transformed into the DL rules as shown below:

$$ps4_1:\ A(ps4) \Rightarrow \text{OBL}\, rel3(X)$$
$$ps4_2:\ A(ps4), \text{OBL}\, rel3, \neg rel3(X), violate(ps5) \Rightarrow \text{OBL}\, rel4(X)$$

5.3.2 *Case 2: Referenced element is a rule*

Instead, if the referenced element is a rule, then for the case of violation, we have to verify that the rule referenced is either (i) inapplicable, i.e., there is a literal in its antecedent that is not provable; or (ii) the immediate consequent of the rule is defeated or overruled by a conflicting conclusion. While for the case of compliance, we have to verify that the referenced rule is applicable and the immediate consequent of the rule is provable[12].

Definition 6
Let $D = (F, R, >)$ be a defeasible theory. $R^b \subseteq R$ (respectively, $R^h \subseteq R$) denotes the set of rules that contains at least one deontic element in their body (head).

[12] In this paper, we consider only the case of weak compliance and weak violation, and verify only the first (modal) literal that appears in the head of the rule. However, the method proposed here can be extended easily to support the verification of the cases of strong compliance (Hashmi *et al.* 2016) and strong violation (Governatori *et al.* 2011).

https://doi.org/10.1017/S1471068418000339 Published online by Cambridge University Press

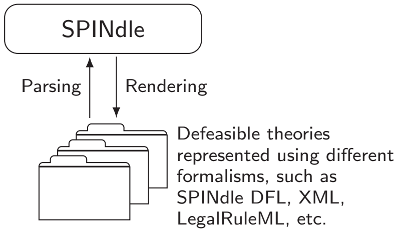

Fig. 3. Theory parsing and rendering process.

Definition 7 (*Rule status*)
Let $D = (F, R, >)$ be a defeasible theory, and let Σ be the language of D. For every $r \in R^b$, r_c denotes the rule referenced by the deontic element (`<lrml:Compliance>` or `<lrml:Violation>`). We define $verifyBody(D) = (F, R', >')$, where

$$
\begin{aligned}
R' = R \setminus R_b \cup \{\, & r_c^+ : A(r_c) \Rightarrow inf(r_c), \\
& r_c^- : \Rightarrow \neg inf(r_c), \\
& r_{cv}^- : \neg inf(r_c) \Rightarrow violation(r_c), \\
& r_{cc}^+ : inf(r_c), comply(C(r_c, 1)) \Rightarrow compliance(r_c), \\
& r_{cv}^+ : inf(r_c), violate(C(r_c, 1)) \Rightarrow violation(r_c) \,\} \\
>' = > \cup \{ & r_c^+ > r_c^- \}
\end{aligned}
$$

For each r_c, $inf(r_c)$, $\neg inf(r_c)$, $compliance(r_c)$, and $violation(r_c)$ are new atoms not in the language of the defeasible theory. $inf(r_c)$ and $\neg inf(r_c)$ are used to determine whether a rule is *in force* (applicable). If r_c is in force, we can then verify whether the first literal that appears at the head of r_c is compliant or violated (represented using the atoms $compliance(r_c)$ and $violation(r_c)$, respectively).

Similar to the case when the referenced object is a literal, depending on where the deontic element is in the rule, we can append the compliance and violation atoms to the body and head of the rule directly. However, unlike the case where the reference element is a literal, this time we can append the atoms required directly without any transformation.

5.4 *Implementation*

The above transformations can be used to translate legal norms represented using LegalRuleML into DL theory that we can reason on. We have implemented the above transformations as an extension to the DL reasoner SPINdle (Lam and Governatori 2009)—an open-source, Java-based DL reasoner. SPINdle supports reasoning on both standard and MDL, such that legal norms represented using LegalRuleML can be parsed into a SPINdle defeasible theory for further processing. Besides, we also implemented a theory renderer so that defeasible theories in SPINdle can also be exported into LegalRuleML documents through a rendering process, as depicted in Figure 3.

To get the idea of how the transformations work, various tests have been carried out to compare the performance of the LegalRuleML theory parser and renderer

https://doi.org/10.1017/S1471068418000339 Published online by Cambridge University Press

with the SPINdle DFL theory parser and render (Lam 2014), respectively. All source code (including SPINdle) in the experiments is compiled using the Java SDK 1.8 without any optimization flags. The times and memory usage presented in the experiments are those measured by the system functions supported by Java Virtual Machine, and was performed on the same lightly loaded Intel Core i5 (3.5 GHz) machine operating under macOS 10.13 with 16 GB main memory. Each timing and memory usage datum is the mean of several executions. There is no substantial variation among the executions, except as noted. Time and memory consumed exclude the latency and overhead caused by SPINdle initialization.

The experiments are based on the Telecommunication Consumer Protections Code described in Governatori *et al.* (2016) which consists of 6 constitutive statements, 78 prescriptive statements, and 10 override statements, and transformed into a defeasible theory with 6 strict rules, 78 defeasible rules, and 10 superiority relations (with 121 literals). In order to further evaluate the scalability of the LegalRuleML theory parser and renderer, we have created a set of synthetic theories by duplicating the set of (all) statements in the original theory and renamed their keys.

The experiments were carried out as follows. The Telecommunication Consumer Protections Code theory represented using LegalRuleML first parsed using the LegalRuleML theory parser into a SPINdle defeasible theory. Then, the generated defeasible theory is transformed back into the LegalRuleML formalism to measure the performance of the rendering process. Note that the LegalRuleML document generated based on the rendering process will be based only on the information available from the defeasible theory in SPINdle, which might not be the same as the one that we used as input. This is due to the fact that SPINdle accepts defeasible theory represented using different formalisms as input (as long as a theory parser associated with the formalism is available). However, as LegalRuleML is essentially more expressive and support more features than the SPINdle defeasible theory, some information may be lost and cannot be captured during the theory generation phase. Hence, the theory generated during the rendering process can only be based on the information available from the defeasible theory, i.e., the set of rules and the penalties/reparations information. Other details, such as the metadata of the norms and information about deontic elements (such as violation and compliance, etc.) will be lost during the rendering process.

Figures 4 and 5 show the performance measured for executing the test theories. As can be seen from the graphs, due to its complexity, it is clear that the LegalRuleML theory parser and renderer, in general, consume more time and memory than the SPINdle DFL theory parser and renderer when parsing and rendering a defeasible theory, respectively. The only exception is on the memory consumption when exporting defeasible theories from SPINdle that the two formalisms consume more-or-less the same amount of memory.

This is understandable as LegalRuleML is more expressive and complex than the SPINdle DFL language, which requires more time to parse and analyze the internal structure of the document. As it is generally the case, there is always a trade-off between expressiveness and efficiency. Even with the size of the deontic theory that we are using, there is already a performance gap between LegalRuleML and the

https://doi.org/10.1017/S1471068418000339 Published online by Cambridge University Press

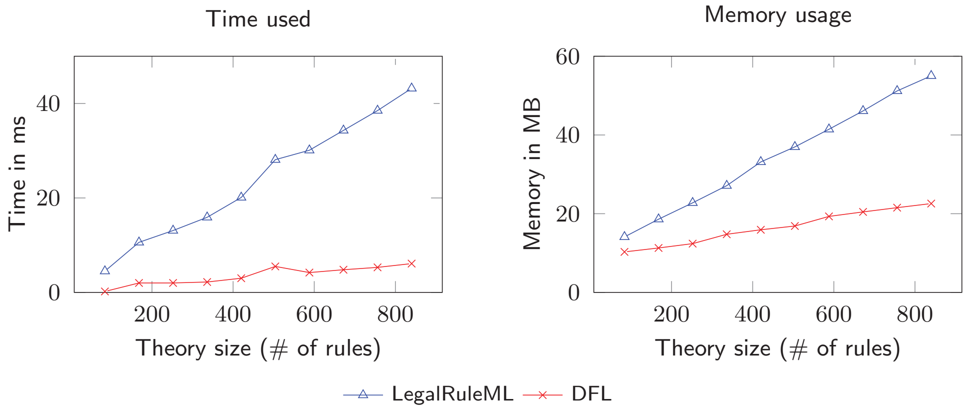

Fig. 4. Performance measurements: Theory parsing.

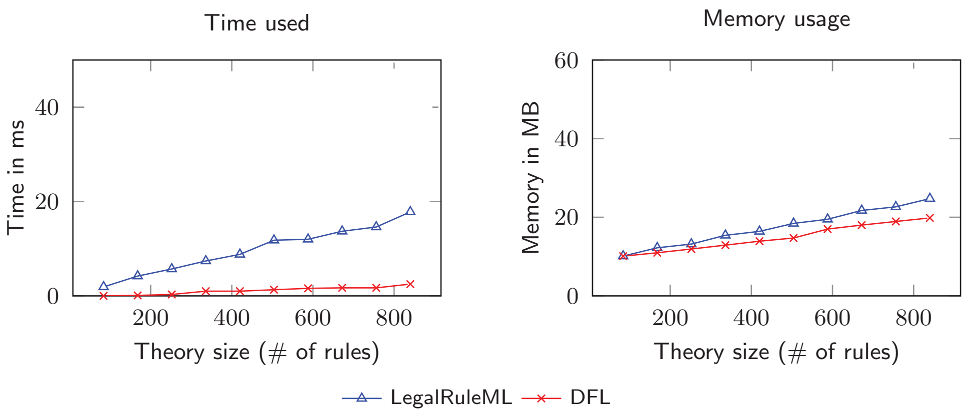

Fig. 5. Performance measurements: Theory rendering.

SPINdle DFL language when parsing the defeasible theory. And the same applies, as well, to the rendering process.

In terms of scalability, except some minor fluctuations which may possibly be caused by the operating system or the Java Virtual Machine (such as heap allocation, page swapping, etc.), both theory parser and renderer perform almost linearly with respect to the size of the theories.

As a remark, the transformation above conforms with the current version of the LegalRuleML specifications (Athan *et al.* 2013). However, it should be noted that strange results may appear if a `<lrml:Violation>` (or `<lrml:Compliance>`) element appears at the head of a statement (i.e., the `<ruleml:then>` part of a statement). For instance, consider the case where the deontic element `<lrml:Violation>` appears as the only element at the head of a statement. Then, it will be transformed into a rule with no head literal, which is not correct. In the light of this, we believe that additional restriction(s) should be added to the LegalRuleML specification in order to avoid this situation.

https://doi.org/10.1017/S1471068418000339 Published online by Cambridge University Press

6 Related work

The research in the area of e-contracting, business process compliance, and automated negotiation systems has evolved over the last few years. Several new modeling languages have been proposed and improvements have been made on the existing ones. On the basis of these modeling languages, different taxonomies and semantics of business rules have been developed (Gordon *et al.* 2009), and transformations techniques have emerged facilitating the reasoning process with these languages.

ContractLog (Paschke *et al.* 2005) is a rule-based framework for monitoring and execution of service level agreements. It combines rule-based representation of service level agreements using Horn clauses and meta programming techniques alternative to contracts defined in natural language or pure programming implementations in programming languages. A rule-based technique called *SweetDeal* for representing business contracts that enables the software agent to automatically create, negotiate, evaluate, and execute the contract provisions with high degree of modularity is discussed in Grosof and Poon (2012). Their technique builds upon situated courteous logic programs knowledge representation in RuleML, and incorporates the process knowledge descriptions whose ontologies are represented in DAML+OIL[13]. DAML+OIL representations allow handling more complex contracts with behavioral provisions that might arise during the execution of contracts. The former has to rely upon multiple formalisms to represent various types of service level agreement rules, e.g., Horn logic, event calculus, description logic—whereas the latter does not consider normative effects (i.e., the approach is unable to differentiate various types of obligations such as achievement, maintenance, and permissions).

SBVR (OMG 2008b) is an Object Management Group standard to represent and fomalize business ontologies, including business rules, facts, and business vocabularies. It provides the basis for detailed formal and declarative specifications of business policies and includes deontic operators to represent deontic concepts, e.g., obligations, permissions, etc. Also, it uses the controlled natural languages to represent legal norms (Gordon *et al.* 2009); however, the standard has some shortcomings as the semantics for the deontic notions is underspecified. This is because SBVR is based on classical first-order logic, which is not suitable to represent deontic notions and conflicts because first-order logic has no conceptual relevance to the legal domain (Herrestad 1991; Hashmi and Governatori 2017). Also, it cannot handle contrary-to-duty obligations as these cannot be represented by standard deontic logic (see Carmo and Jones (2002) for details).

The Legal Knowledge Interchange Format (ESTRELLA Project 2008), on the other hand, is an XML-based interchange format language that aims to provide an interchangeable format to represent legal norms in a broad range of application scenarios—especially in the context of semantic web. Legal Knowledge Interchange Format uses XML schemas to represent theories and arguments derived from

[13] DAM+OIL Reference: `http://www.w3.org/TR/daml+oil-reference/`

https://doi.org/10.1017/S1471068418000339 Published online by Cambridge University Press

theories, where a theory in Legal Knowledge Interchange Format is a set of axioms and defeasible inference rules. In addition to these, there are other XML-based rule interchange format languages, e.g., Semantic Web Rule Language (Horrocks *et al.* 2004), RIF (W3C RIF Working Group 2005), WSMO (Roman *et al.* 2005), and OWL for Services (The OWL Services Coalition 2006) to name but few, see Gordon *et al.* (2009) for more details on the strengths and weaknesses of these languages.

Antoniou *et al.* (2009) presented a model and deontic-based system for representing and reasoning policies in multi-agent systems. Their logical framework extends modal logics with modalities such as *knowledge*, *intention*, *agency*, and *obligations*, and, similar to the approach discussed above, the reasoning part is completed through simulated meta programming. Furthermore, their model also support standard RDF and RDF schema, which enable their approach to be compatible with other semantic web technologies.

Kontopoulos *et al.* (2011), on the other hand, extended DR-DEVICE (Bassiliades *et al.* 2004) with the capabilities to reason on MDL rule bases. Their approach is based on a RuleML like formalism with the support of interactions between modalities, which may be useful from semantic web perspective where a greedy agent can override its contractual obligations and perform what it intends to do. Similar to our work proposed here, their approach is based on a series of transformations that will progressively transform the rule-based into a DL theory (for reasoning purpose) and additional elements, such as defeaters, may be added to the theory to resolve issues such as *modality inclusion*. However, from legal perspective, it may not be the same as it is necessary that each party complies with the obligations that are stated in the agreement. Hence, we have not included this in our framework. Nevertheless, as a note, such features can easily be supported as it is a built-in feature in SPINdle.

Steen *et al.* (2010) proposed an approach that can automatically transform business rules specifications written using SBVR into optimized business process models modeled using BPMN (OMG 2008a) and a domain model represented using UML (OMG 2000). Later, through utilizing a combination of techniques from cognitive linguistics, knowledge configuration, and model-driven engineering, Selway *et al.* (2015) proposed an approach that automatically transforms a business specification into a formal SBVR model of vocabulary and business rules specification. However, the transformations proposed by Steen *et al.* (2010) are correct by construction but have not been formally verified. In particular, the transformations related to the process meta models are generated from the extracted SBVR specifications. Besides, even though both approaches enable the generation of suitable formal models from business specifications, the rules transformations and generated models do not include any information on various types of rules as the deontic notions are underspecified in SBVR.

Wyner and Governatori (2013) used C&C/Boxer (Bos 2008) to automatically translate normative clauses into semantics representations, and compared the results with the logical representation that they created manually (using DL), which narrowed the gap between natural language sources materials and formal, machine-processable representations. However, it is not clear how C&C/Boxer can be used as an abstract representation that is required by C&C/Boxer. In contrast, Baget

https://doi.org/10.1017/S1471068418000339 Published online by Cambridge University Press

et al. (2015) discuss techniques for transforming existential rules into Datalog^{+},[14], RuleML, and OWL 2 formats. For the transformation from Datalog^{+} into RuleML, the authors used a fragment of Deliberation RuleML 1.01, which includes positive facts, universally quantified implications, equality, falsity (and conjunctions) in the heads of implications. Whereas Vojír *et al*. (2013) transforms the association rules into Drool Rule Language using Lisp-Miner[15], Kamada *et al*. (2010) proposes a model-driven architecture-based model to transform SBVR compliant business rules extracted from business contracts of services to compliant executable rules in Formal Contract Language (Governatori and Milosevic 2005). However, the former's transformation is limited only to existential rules, while the latter captures only the business rules (SBVR bears only business rules), which may or may not have legal standings. While, LegalRuleML represents legal standings, the LegalRuleML's temporal notions of enforceability, efficacy, and applicability cannot be represented with SBVR. In contrast, the approach proposed in this paper enables the translation of defeasible expressions, and various *deontic concepts* including the notion of *penalty* and *chain of reparations*.

7 Conclusions

In this paper, we have proposed a transformation such that (legal) norms represented using LegalRuleML can be transformed into DL which provides us a method for modeling business contracts and reasoning about them in a declarative way. While LegalRuleML aims at providing specifications to the legal norms that can be represented in a machine-readable format, the major impedance now is the lack of dedicated and reliable infrastructure that can provide support to such capability.

As a future work, we are planning to incorporate our technique into some smart-contract enabled systems, such as Ethereum (Wood 2014). This will extend its language such that, instead of using programming logics, users can define their (smart-)contracts in a declarative manner. Besides, we also plan to examine the feasibility of the presented approach in the context of cyber-physical systems, in particular, translation of standards-based regulations for the verification of planned and executed processes for automotive software development.

Since legal documents tend to continuously evolve meaning that new rules are added or removed. The addition of new rules into legal documents may introduce new rule types with varied set of granularity which may increase the complexity of reasoning process. Currently, we have applied our transformation approach to legal statements that appear in very common settings. We plan to continue our experiments validating our transforming approach with larger theories and rules sets with varied degree of complexity in very concentrated settings.

[14] Datalog^{+}: a sub-language of RuleML `http://wiki.ruleml.org/index.php/Rule-Based_Data_Access#Datalog.2B.2F-`

[15] Lisp-Miner: `http://lispminer.vse.cz`

https://doi.org/10.1017/S1471068418000339 Published online by Cambridge University Press

References

ANTONIOU, G. 2004. A discussion of some intuitions of defeasible reasoning. In *Proc. of the 3rd Hellenic Conference on AI: Methods and Applications of Artificial Intelligence*, G. A. Vouros and T. Panayiotopoulos, Eds. Springer, Berlin/Heidelberg, Samos, Greece, 311–320.

ANTONIOU, G. AND BIKAKIS, A. 2007. DR-Prolog: A system for defeasible reasoning with rules and ontologies on the semantic web. *IEEE Transactions on Knowledge and Data Engineering 19,* 2, 233–245.

ANTONIOU, G., BILLINGTON, D., GOVERNATORI, G. AND MAHER, M. J. 2000. A flexible framework for defeasible logics. In *Proc. of the 17th National Conference on Artificial Intelligence*. AAAI Press/The MIT Press, Austin, Texas, USA, 405–410.

ANTONIOU, G., BILLINGTON, D., GOVERNATORI, G. AND MAHER, M. J. 2001. Representation Results for Defeasible Logic. *ACM Transactions on Computational Logic 2,* 2, 255–286.

ANTONIOU, G., DIMARESIS, N. AND GOVERNATORI, G. 2009. A modal and deontic defeasible reasoning system for modelling policies and multi-agent systems. *Expert Systems with Applications 36,* 2, 4125–4134.

ATHAN, T., BOLEY, H., GOVERNATORI, G., PALMIRANI, M., PASCHKE, A. AND WYNER, A. 2013. OASIS LegalRuleML. In *Proc. of the 14th International Conference on Artificial Intelligence and Law*. ACM, Rome, Italy, 3–12.

ATHAN, T., GOVERNATORI, G., PALMIRANI, M., PASCHKE, A. AND WYNER, A. 2015. LegalRuleML: Design principles and foundations. In *Proc. of the 11th Reasoning Web Summer School*, W. Faber and A. Paschke, Eds. Springer, Berlin, Germany, 151–188.

BAGET, J., GUTIERREZ, A., LECLÈRE, M., MUGNIER, M., ROCHER, S. AND SIPIETER, C. 2015. Datalog+, RuleML and OWL 2: Formats and translations for existential rules. In *Proc. of the RuleML 2015 Challenge*, N. Bassiliades, P. Fodor, A. Giurca, G. Gottlob, T. Kliegr, G. J. Nalepa, M. Palmirani, A. Paschke, M. Proctor, D. Roman, F. Sadri, and N. Stojanovic, Eds. CEUR Workshop Proceedings, Berlin, Germany.

BASSILIADES, N., ANTONIOU, G. AND VLAHAVAS, I. 2004. DR-DEVICE: A defeasible logic system for the semantic web. In *Proc. of the 2nd International Workshop on Principles and Practice of Semantic Web Reasoning*, H. J. Ohlbach and S. Schaffert, Eds. Springer-Verlag, St Malo, France, 134–148.

BOS, J. 2008. Wide-coverage semantic analysis with boxer. In *Proc. of the 2008 Conference on Semantics in Text Processing*. Association for Computational Linguistics, Stroudsburg, PA, USA, 277–286.

CALÌ, A., GOTTLOB, G. AND LUKASIEWICZ, T. 2012. A general datalog-based framework for tractable query answering over ontologies. *Web Semantics: Science, Services and Agents on the World Wide Web 14*, 57–83.

CARMO, J. AND JONES, A. J. I. 2002. Deontic logic and contrary-to-uties. In *Handbook of Philosophical Logic*, 2nd ed., Vol. 8, D. Gabbay and F. Guentner, Eds. Kluwer, Dordrecht, 265–343.

DEAGUSTINI, C. A., MARTINEZ, M. V., FALAPPA, M. A. AND SIMARI, G. R. 2015. On the Influence of incoherence in inconsistency-tolerant semantics for datalog$^{\pm}$. In *Proc. of the Joint Ontology Workshops*, O. Papini, S. Benferhat, L. Garcia, M.-L. Mugnier, E. Fermé, T. Meyer, R. Wassermann, K. Baclawski, A. Krisnadhi, P. Klinov, S. Borgo, O. Kutz, and D. Porello, Eds. CEUR Workshop Proceedings, Buenos Aires, Argentina.

ESTRELLA PROJECT. 2008. The Legal Knowledge Interchange Format (LKIF). Deliverable 4.1, European Commission.

FARRELL, A. D. H., SERGOT, M. J., SALLE, M., BARTOLINI, C., TRASTOUR, D. AND CHRISTODOULOU, A. 2004. Performance monitoring of service-level agreements for utility computing using the event calculus. In *Proc. of the 1st IEEE International Workshop on Electronic Contracting*. San Diego, CA, USA, 17–24.

https://doi.org/10.1017/S1471068418000339 Published online by Cambridge University Press

Goedertier, S. and Vanthienen, J. 2006. Designing compliant business processes with obligations and permissions. In *Business Process Management Workshops*, J. Eder and S. Dustdar, Eds. Springer, Heidelberg, Vienna, Austria, 5–14.

Gordon, T. F., Governatori, G. and Rotolo, A. 2009. Rules and norms: Requirements for rule interchange languages in the legal domain. In *Proc. of the 2009 International Symposium on Rule Interchange and Applications*. Springer, Heidelberg, Las Vegas, Nevada, USA, 282–296.

Governatori, G. 2005. Representing business contracts in RuleML. *International Journal of Cooperative Information Systems 14,* 2–3, 181–216.

Governatori, G., Hashmi, M., Lam, H.-P., Villata, S. and Palmirani, M. 2016. Semantic business process regulatory compliance checking using LegalRuleML. In *Proc. of the 20th International Conference on Knowledge Engineering and Knowledge Management*, E. Blomqvist and F. Vitali, Eds. Springer, Bologna, Italy.

Governatori, G., Maher, M. J., Antoniou, G. and Billington, D. 2004. Argumentation semantics for defeasible logic. *Journal of Logic and Computation 14,* 5, 675–702.

Governatori, G. and Milosevic, Z. 2005. Dealing with contract violations: Formalism and domain specific language. In *Proc. of the 9th IEEE International Enterprise Distributed Object Computing Conference*. IEEE Computer Society, Enschede, the Netherlands, 46–57.

Governatori, G., Olivieri, F., Scannapieco, S. and Cristani, M. 2011. Designing for compliance: Norms and goals. In *Proc. of the 5th International Symposium on Rule-Based Modeling and Computing on the Semantic Web*, F. Olken, M. Palmirani, and D. Sottara, Eds. Springer, Berlin, Heidelberg, Ft. Lauderdale, Florida, USA, 282–297.

Governatori, G. and Rotolo, A. 2004. Defeasible logic: Agency, intention and obligation. In *Proc. of the 7th International Workshop on Deontic Logic in Computer Science*, A. Lomuscio and D. Nute, Eds. Springer-Verlag, Madeira, Portugal, 114–128.

Governatori, G. and Rotolo, A. 2006. Logic of violations: A gentzen system for reasoning with contrary-to-duty obligations. *Australasian Journal of Logic 4*, 193–215.

Governatori, G. and Rotolo, A. 2008a. A computational framework for institutional agency. *Artificial Intelligence and Law 16,* 1, 25–52.

Governatori, G. and Rotolo, A. 2008b. BIO logical agents: Norms, beliefs, intentions in defeasible logic. *Autonomous Agents and Multi-Agent Systems 17,* 1, 36–69.

Grosof, B. and Poon, T. C. 2012. SweetDeal: Representing agent contracts with exceptions using XML rules, ontologies, and process descriptions. In *the 12th International World Wide Web Conference*. ACM, Budapest, Hungary, 340–349.

Grosof, B. N. 2004. Representing e-commerce rules via situated courteous logic programs in RuleML. *Electronic Commerce Research and Applications 3,* 1, 2–20.

Hashmi, M. and Governatori, G. 2017. Norms modeling constructs of business process compliance management frameworks: A conceptual evaluation. *Artificial Intelligence and Law*. URL: `https://doi.org/10.1007/s10506-017-9215-8` [*in print*].

Hashmi, M., Governatori, G. and Wynn, M. T. 2016. Normative requirements for regulatory compliance: An abstract formal framework. *Information Systems Frontiers 18,* 3, 429–455.

Hecham, A., Croitoru, M. and Bisquert, P. 2017. Argumentation-based defeasible reasoning for existential rules. In *Proc. of the 16th Conference on Autonomous Agents and Multiagent Systems*, K. Larson, M. Winikoff, S. Das and E. H. Durfee, Eds. International Foundation for Autonomous Agents and Multiagent Systems. Sã Paulo, Brazil, 1568–1569.

Herrestad, H. 1991. Norms and Formalization. In *Proc. of the 3rd International Conference on Artificial Intelligence and Law*. ACM, Oxford, England, 175–184.

Horrocks, I., Patel-Schneider, P. F., Boley, H., Tabet, S., Grosof, B. and Dean, M. 2004. SWRL: A semantic web rule language. URL: `https://www.w3.org/Submission/SWRL/`. Accessed 7 Feb 2017.

https://doi.org/10.1017/S1471068418000339 Published online by Cambridge University Press

Kamada, A., Governatori, G. and Sadiq, S. 2010. Transformation of SBVR compliant business rules to executable FCL rules. In *Rule-Based Modeling and Computing on the Semantic Web*, M. Dean, J. Hall, A. Rotolo and S. Tabet, Eds. Springer, Berlin/Heidelberg, Washington, DC, USA, 153–161.

Kontopoulos, E., Bassiliades, N., Governatori, G. and Antoniou, G. 2011. A modal defeasible reasoner of deontic logic for the semantic web. *International Journal on Semantic Web and Information Systems 7,* 1, 18–43.

Lam, H.-P. 2014. *SPINdle User Guide*. CSIRO, Australia. Accessed 15 March 2018. URL: `http://spindle.data61.csiro.au/spindle/documentation.html`.

Lam, H.-P. and Governatori, G. 2009. The making of SPINdle. In *Proc. of the International Symposium on Rule Interchange and Applications*, A. Paschke, G. Governatori and J. Hall, Eds. Springer-Verlag, Las Vegas, Nevada, 315–322.

Lam, H.-P. and Governatori, G. 2011. What are the necessity rules in defeasible reasoning? In *Proc. of the 11th International Conference on Logic Programming and Nonmonotonic Reasoning*. Springer, Berlin/Heidelberg, Vancouver, BC, Canada, 187–192.

Lam, H.-P. and Governatori, G. 2013. Towards a model of UAVs navigation in urban canyon through Defeasible Logic. *Journal of Logic and Computation 23,* 2, 373–395.

Lam, H.-P., Governatori, G. and Riveret, R. 2016. On ASPIC$^+$ and defeasible logic. In *Proc. of the 6th International Conference on Computational Models of Argument*, P. Baroni, T. F. Gordon, T. Scheffler and M. Stede, Eds. IOS Press, Potsdam, Germany, 359–370.

Lam, H.-P., Hashmi, M. and Scofield, B. 2016. Enabling reasoning with LegalRuleML. In *Proc. of the 10th International Web Rule Symposium*, J. J. Alferes, L. Bertossi, G. Governatori, P. Fodor, and D. Roman, Eds. Springer International Publishing, Stony Brook, NY, USA, 241–257.

Maher, M. J. 2001. Propositional defeasible logic has linear complexity. *Theory and Practice of Logic Programming 1,* 6, 691–711.

Maher, M. J., Rock, A., Antoniou, G., Billington, D. and Miller, T. 2001. Efficient Defeasible Reasoning Systems. *International Journal on Artificial Intelligence Tools 10,* 4, 483–501.

Martin, D., Burstein, M., Hobbs, J., LAssila, O., McDermott, D., McIlraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., Sirin, E., Srinvasan, N. and Sycara, K. 2004. OWL-S: Semantic markup for web services. URL: `https://www.w3.org/Submission/OWL-S/`. Accessed 7 Feb 2017.

Martinez, M. V., Deagustini, C. A. D., Falappa, M. A. and Simari, G. R. 2014. Inconsistency-tolerant reasoning in Datalog$^\pm$ ontologies via an argumentative semantics. In *Proc. of the 14th Ibero-American Conference on AI*, A. L. Bazzan and K. Pichara, Eds. Springer International Publishing, Santiago de Chile, Chile, 15–27.

Modgil, S. and Prakken, H. 2013. A general account of argumentation with preferences. *Artificial Intelligence 195*, 361–397.

Modgil, S. and Prakken, H. 2014. The ASPIC$^+$ framework for structured argumentation: A tutorial. *Argument & Computation 5,* 1, 31–62.

Nute, D. 2001. Defeasible logic: Theory, implementation and applications. In *Proc. of the 14th International Conference on Applications of Prolog*. Springer, Berlin, Tokyo, Japan, 151–169.

OASIS LegalRuleML TC. 2013. OASIS LegalRuleML. URL: `https://www.oasis-open.org/committees/legalruleml`. Accessed 7 Feb 2017.

OMG 2000. *Unified Modeling Language (UML)*. OMG. URL: `http://www.omg.org/spec/UML/`. Accessed 7 Feb 2017.

https://doi.org/10.1017/S1471068418000339 Published online by Cambridge University Press

OMG 2008a. *Business Process Model And Notation (BPMN)*. OMG. URL: http://www.omg.org/spec/BPMN/. Accessed 7 Feb 2017.

OMG 2008b. *Semantics of Business Vocabulary And Rules (SBVR)*. OMG. URL: http://www.omg.org/spec/SBVR. Accessed 7 Feb 2017.

Palmirani, M., Governatori, G., Athan, T., Boley, H., Paschke, A. and Wyner, A. 2015. LegalRuleML Core Specifications. URL: http://docs.oasis-open.org/legalruleml/legalruleml-core-spec/v1.0/legalruleml-core-spec-v1.0.html. Accessed 7 Feb 2017.

Paschke, A., Bichler, M. and Dietrich, J. 2005. ContractLog: An approach to rule based monitoring and execution of service level agreements. In *Proc. of the 1st International Conference on Rules and Rule Markup Languages for the Semantic Web*. Springer, Berlin, Heidelberg, Galway, Ireland, 209–217.

Pesic, M. and Aalst, W. M. P. 2006. A declarative approach for flexible business processes management. In *BPM Workshops 2006*. Springer, Berlin, Heidelberg, Austria, 169–180.

Prakken, H. 2010. An abstract framework for argumentation with structured arguments. *Argument & Computation 1,* 2, 93–124.

Roman, D., Keller, U., Lausen, H., de Bruijn, J., Lara, R., Stollberg, M., Polleres, A., Feier, C., Bussler, C. and Fensel, D. 2005. Web service modeling ontology. *Applied Ontology 1,* 1, 77–106.

RuleML Inc. 2000. RuleML: The rule markup initiative. URL: http://www.ruleml.org. Accessed 7 Feb 2017.

Searle, J. R. 1997. *The Construction of Social Reality*. Free Press, New York, NY, USA.

Selway, M., Grossmann, G., Mayer, W. and Stumptner, M. 2015. Formalising natural language specifications using a cognitive linguistic/configuration based approach. *Information Systems 54*, 191–208.

Song, I. 2008. Design agent chips. Ph.D. thesis, The University of Queensland, St. Lucia, Queensland, Australia.

Steen, B., Pires, L. F. and Iacob, M.-E. 2010. Automatic generation of optimal business processes from business rules. In *Proc. of the 14th IEEE International Enterprise Distributed Object Computing Conference Workshops*. Vitoria, Brazil, 117–126.

The OWL Services Coalition. 2006. OWL-S Specification. URL: http://www.ai.sri.com/daml/services/owl-s/. Accessed 7 Feb 2017.

Vojír, S., Kliegr, T., Hazucha, A., Skrabal, R. and Simunek, M. 2013. Transforming association rules to business rules: EasyMiner meets drools. In *Joint Proc. of the 7th International Rule Challenge, the Special Track on Human Language Technology and the 3rd RuleML Doctoral Consortium*, P. Fodor, D. Roman, D. Anicic, A. Wyner, M. Palmirani, D. Sottara, and F. Lévy, Eds. CEUR Workshop Proceedings, Seattle, USA.

W3C RIF Working Group. 2005. RIF: Rule Interchange Format. URL: https://www.w3.org/standards/techs/rif. Accessed 7 Feb 2017.

Wood, G. 2014. Ethereum: A Secure Decentralised Generalised Transaction Ledger. Accessed 12 March 2016. URL: http://gavwood.com/paper.pdf.

Wyner, A. and Governatori, G. 2013. A study on translating regulatory rules from natural language to defeasible logic. In *Joint Proc. of the 7th International Rule Challenge, the Special Track on Human Language Technology and the 3rd RuleML Doctoral Consortium*, P. Fodor, D. Roman, D. Anicic, A. Wyner, M. Palmirani and D. S. F. Lévy, Eds. CEUR Workshop Proceedings, Seatle, USA.

https://doi.org/10.1017/S1471068418000339 Published online by Cambridge University Press