OASIS LegalRuleML

Tara Athan
Athan Services, USA

Harold Boley
Faculty of Computer Science
University of New Brunswick,
Canada

Guido Governatori
NICTA, Australia

Monica Palmirani
CIRSFID
University of Bologna, Italy

Adrian Pasckhe
Corporate Semantic Web
Freie Universitaet Berlin, Germany

Adam Wyner
Department of Computing Science
University of Aberdeen, UK

ABSTRACT

In this paper we present the motivation, use cases, design principles, abstract syntax, and initial core of LegalRuleML. The LegalRuleML-core is sufficiently rich for expressing legal sources, time, defeasibility, and deontic operators. An example is provided. LegalRuleMLis compared to related work.

General Terms

Languages, Standardization

Keywords

LegalRuleML

1. INTRODUCTION

In this section, we introduce the motivation, requirements, and design principles for LegalRuleML

1.1 Motivation

Legal texts, e.g. legislation, regulations, contracts, and case law, are the source of norms, guidelines, and rules. As text, it is difficult to exchange specific data between parties, to search for and extract structured information within the text, or to automatically process further. Legislators, legal practicioners, business managers are, therefore, impeded from comparing, contrasting, integrating, and reusing the contents of the texts, since any such activities are manual. In the current web-enabled context, where innovative eGovernment and eCommerce applications are increasingly deployed, it becomes essential to provide machine-readable forms (generally in XML) of the contents of the text. In doing so, the general norms and specific procedural rules in legislative documents, the conditions of services and business rules in contracts, and the information about arguments and interpretation of norms in the judgements for case-law would be amenable to such applications.

The ability to have proper and expressive conceptual, machine-readable models of the various and multifaceted aspects of norms, guidelines, and general legal knowledge is a key factor for the development and deployment of successful applications. The LegalRuleML TC, set up inside of OASIS (`www.oasis-open.org`), aims to produce a rule interchange language for the legal domain. Using the representation tools, implementers can structure the contents of the legal texts in a machine-readable format, which then feeds further processes of interchange, comparison, evaluation, and reasoning.

The Artificial Intelligence (AI) and Law communities have converged in the last twenty years on modelling legal norms and guidelines using logic and other formal techniques [Ashley and van Engers, 2011]. Existing methods begin with the analysis of a legal text by a Legal Knowledge Engineer who extracts the norms and guidelines, applies models and a theory within a logical framework, and finally represents the norms using a particular formalism. In the last decade, several Legal XML standards have been proposed to describe legal texts [Lupo et al., 2007] with XML-based rules (RuleML, SWRL, RIF, LKIF, etc.) [Gordon et al., 2009, Gordon, 2008]. At the same time, the Semantic Web, in particular Legal Ontology research combined with semantic norm extraction based on Natural Language Processing (NLP) [Francesconi et al., 2010], has given a strong impetus to the modelling of legal concepts [Boer et al., 2008, Benjamins et al., 2005, Breuker et al., 2006].

Based on this, the work of the LegalRuleML Technical Committee will focus on three specific needs:

1. To close the gap between natural language text description and semantic norm modelling. This is necessary in order to realise an integrated and self-contained representation of legal resources that can be made available on the Web as XML representations [Palmirani et al., 2009] and so foster Semantic Web technologies such as: NLP, Information Retrieval and Extraction (IR/IE), graph representation, as well as Web ontologies and rules.
2. To provide an expressive XML standard for modelling normative rules that satisfies legal domain requirements. This will enable use of a legal reasoning level on top of the ontological layer that aligns with the W3C envisioned Semantic Web stack. This approach seeks also to fill the gap between regulative norms, guidelines and business rules in order to capture and model the processes embedded in them and make the processes usable for business automation [Governatori and Rotolo, 2010, Grosof, 2004].
3. To extend the Linked Open Data [Berners-Lee, 2010] approach to modelling from raw data (acts, contracts, court files, judgments, etc.) to legal concepts and rules along with their functionality and usage. Without rules that apply to legal concepts, legal concepts constitute just a taxonomy [Sartor, 2009].

ICAIL'13 Jun 10-14 2013, Rome, Italy
ACM 978-1-4503-2080-1/13/06.

1.2 Requirements

Specifically, the LegalRuleML work facilitates the following functionalities.

R1) *Support for modelling different types of rules*:

- *Constitutive rules*, which define concepts or constitute activities that cannot exist without such rules (especially Legal definitions such as "property").
- *Prescriptive rules*, which regulate actions by making them obligatory, permitted, or prohibited (especially obligations in contracts).

R2) *Implement isomorphism* [Bench-Capon and Coenen, 1992]. To ease validation and maintenance, there should be a one-to-one correspondence between collections of rules in the formal model and the units of (controlled) natural language text that express the rules in the original legal sources, such as sections of legislation.

R3) *Manage rule reification* [Gordon, 1995]. Rules are objects with properties, such as Jurisdiction, Authority, Temporal attributes [Palmirani et al., 2010, Governatori et al., 2009, 2005]. These elements necessary to enable effective legal reasoning.

R4) *Represent normative effects and values*. There are many normative effects that follow from applying rules [Gordon et al., 2009]. For prescriptive rules we consider deontic effects such as obligations, permissions, prohibitions, and eventually more articulated effects. Usually, legal rules promote social values, e.g. *freedom*, *privacy*, or *efficiency* as well.

R5) *Implement defeasibility* [Gordon, 1995, Prakken and Sartor, 1996, Sartor, 2005]. In the law, where the antecedent of a rule is satisfied by the facts of a case (or via other rules), the conclusion of the rule presumably, but not necessarily, holds. The defeasibility of legal rules breaks down into conflictual and exclusionary relations amongst rules.

R6) *Model legal procedural rules*. Rules not only regulate the procedures for resolving legal conflicts, but also are used for arguing or reasoning about whether or not some action or state complies with other, substantive rules. In particular, rules are required for procedures which regulate methods for detecting violations of the law and for determining the normative effects triggered by norm violations, such as reparative obligations, which are meant to repair or compensate violations. These constructions can give rise to very complex rule dependencies, because the violation of a single rule can activate other (reparative) rules, which in turn, in case of their violation, refer to other rules, and so forth.

1.3 Design Principles

LegalRuleML is based on some main principles.

DP1) *Multiple semantic annotations*: A legal rule may have multiple semantic annotations, where each annotation can represent a different legal interpretation. Each such annotation can appear in a separate annotation block as internal or external metadata. A range of parameters can be set to provide the interpretation, e.g. provenance, applicable jurisdiction, logical interpretation of the rule, and others.

DP2) *Tracking of LegalRuleML creators*: As part of the provenance information, a LegalRuleML document or any of its fragments can be associated with its creators. This is important to manage the authority and trust of the knowledge base and annotations. Among the creators of the document can be the authors of the text, of the knowledge base, of the annotations, as well as the publisher of the document.

DP3) *Linking rules and provisions*: LegalRuleML includes a mechanism, based on IRI, that allows N:M relationships among the rules and the textual provisions: multiple rules are embedded in the same provision, several provisions contribute to the same rule. This mechanism may be managed in the metadata block for permitting extensible management, avoiding redundancy in the IRI definition, and avoiding errors in the associations.

DP4) *Temporal management*: The universe of discourse to which LegalRuleML applies contains a variety of entities: provisions references, rules, applications of rules and physical entities. All of these entities exist and change in time, and their histories interact in complicated ways. LegalRuleML must represent these temporal issues in an unambiguous fashion. In particular, a rule has a range of parameters which can vary over time such as its status (e.g. strict, defeasible, defeater), its validity (e.g. repealed, annulled, suspended) and its jurisdiction (e.g. only in EU, only in US). In addition, a rule has a variety of temporal aspects such as internal constituency of the action, the time of assertion of the rule, the efficacy, enforcement, and so on.

DP5) *Formal ontology reference*: LegalRuleML is independent from any legal ontology and logic framework. However it includes a mechanism, based on IRIs, for pointing to reusable classes of a specified external ontology.

DP6) *LegalRuleML is based on RuleML*: LegalRuleML reuses and extends concepts and syntax of RuleML (`http://ruleml.org/`) wherever possible, and also adds novel annotations. RuleML includes also Reaction RuleML.

DP7) *Mapping*: LegalRuleML may leverage existing capabilities, including Linked Data and XACML, by mapping portions of LegalRuleML to other syntactic representations.

Additional general design principles considered are:

DP8) *Minimality*. The language provides only a small set of needed language constructs, i.e., the same meaning cannot be expressed by different language constructs.

DP9) *Referential transparency*. The same language construct always expresses the same semantics regardless of the context in which it is used.

DP10) *Orthogonality*. Pairwise independent language constructs are given, thus permitting their meaningful systematic combination.

2. LINKING LEGAL INFORMATION

In legal texts, one may find a variety of links between information within the text as well as between the textual source and a machine-readable representation. For such links and to satisfy the requirements and design principles, we introduce links to textual sources, to agents, authority, jurisdications, to time and events, and to contextual associations.

First of all, following Kelsen [1960], a norm is the meaning of an act of will by which a certain behavior is commanded, permitted or authorized. Norms are usually expressed in written using legal textual provisions (sentences, articles, paragraphs of legal acts). A (legal) rule is a logic formalization of the textual provisions in the form of antecedent and consequent. Sometime several textual provisions determine a single rule or a single provision includes multiple rules.

In the following and in keeping with (DP6), rules are represented as in RuleML, which we assume in the presentation.

2.1 Sources and Isomorphism

One of the main problems of legal rule modelling is to maintain isomorphism — the connection between the formal rules and the legally binding textual statements modelled by the rules. As the legal source changes over time, the formal rules need to be updated, and usually there is no automatic mechanism to correlate and track such modifications. Moreover, it is fundamental for the judge, citizen, or legal operator to verify that the formalisations of the norms as modelled by legal knowledge engineers are conceptually consistent with the original legally binding texts. For this reason LegalRuleML includes a component to help manage the isomorphism at a fine granularity and to represent modifications with respect to the temporal dimension.

Several blocks of XML are dedicated to annotate the original legal sources and to connect them to rules, so permitting an N:M relationship (e.g. many rules in one textual provision, many textual provision for only one rule). There are blocks for sources and blocks that associate sources with rules, assuming references to rules such as `rule1`. <References> is the block dedicated to record non-IRI based identifier sources and the attribute `refIDSystemName` is able to annotate the naming convention used:

```
<lrml:References>
  <lrml:Reference refersTo="ref1"
    refID="/us/USCode/eng@/main#title17-sec504
-rclsc-pnt1"
    refIDSystemName="AkomaNtoso2.0-2012-10"/>
</lrml:References>
```

<LegalSource> is the block dedicated to record the IRI based identifier sources:

```
<lrml:LegalSources>
  <lrml:LegalSource key="ref2"
    sameAs="http://www.law.cornell.edu/uscode/text/
17/504#psection-1"/>
</lrml:LegalSources>
```

Finally the block <Association> links sources/references with the relatives rules, thus implementing the N:M relationship. For one source to many rules, we have:

```
<lrml:Association>
  <lrml:appliesSource keyref="#ref1"/>
  <lrml:toTarget keyref="#rule1"/>
  <lrml:toTarget keyref="#rule2"/>
</lrml:Association>
```

For one rule with multiple sources, we have the following, where `rule1` is connected to `ref1` (above) and to `ref2` (below):

```
<lrml:Association>
  <lrml:appliesSource keyref="#ref2"/>
  <lrml:toTarget keyref="#rule1"/>
</lrml:Association>
```

2.2 Agents and Authority

We must represent the parameters concerning the provenance and the authoritativeness of the rules: the author of the rule formalization and the associated the authority. To indicate such parameters, we have blocks for the specfication of agents and authorities:

```
<lrml:Agents>
  <lrml:Agent key="aut1"
    sameAs="&unibo;/person.owl#m.palmirani"/>
  <lrml:Agent key="aut2"
    sameAs="&unibo;/person.owl#g.governatori"/>
</lrml:Agents>
<lrml:Authorities>
  <lrml:Authority key="congress"
    sameAs="&unibo;/org.owl#congress">
    <lrml:type iri="&lrmlv;Legislature"/>
  </lrml:Authority>
</lrml:Authorities>
```

In Section 2.4, we introduce other parameters to associate the proper author, authorities, jurisdiction, source, and other characteristics to the rule using <Context> block. Here, we have neutrally and simply modelled the agents and the authorities. In this way, it is possible, for example, to model a norm that follows two different jurisprudential doctrines (e.g. Positivism and Realism) by qualifying the rule with respect to multiple authorities or agents.

2.3 Time and Events

Legal texts are often amended over time following the evolution of the society or judicial system. The norms and the related rules are valid in a particular interval of time and with respect to three main legal axes: entry into force, efficacy, and applicability. In this section, we model the external temporal dimensions of the norms (when the norm is valid) and not the complex events that are the content of the textual provision (e.g., when a person is to present a tax application). Therefore, we only model the events, intervals and temporal parameters that define the period of validity of the rules. Moreover, in keeping with the sources, it is important to link the temporal parameters to any part of a rule (e.g. `atom`, `rel`, `ind`, `if`, `then`, etc.) with a very fine granularity.

```
<lrml:TimeInstants>
  <ruleml:Time key="t1">
    <ruleml:Data xsi:type="xs:dateTime">
      1978-01-01T00:00:00
    </ruleml:Data>
  </ruleml:Time>
</lrml:TimeInstants>
```

The events are combined in intervals according with the legal temporal situation that is modelled, e.g. enforceability, efficacy, applicability.

```
<lrml:TemporalCharacteristics>
  <lrml:TemporalCharacteristic>
    <lrml:forRuleStatus
      iri="&lrmlv;#Efficacious"/>
    <lrml:hasStatusDevelopment
      iri="&lrmlv;#Starts"/>
    <lrml:atTimeInstant keyref="#t1"/>
  </lrml:TemporalCharacteristic>
  <lrml:TemporalCharacteristic>
    <lrml:forRuleStatus
      iri="&lrmlv;#Efficacious"/>
    <lrml:hasStatusDevelopment
      iri="&lrmlv;#End"/>
    <lrml:atTimeInstant keyref="#t2"/>
  </lrml:TemporalCharacteristic>
</lrml:TemporalCharacteristics>
```

The block with key `e1-b` can be reused in any part of the rule formalization avoiding redundancy in the legal situation definition.

2.4 Context Associations

A rule may be associated with a variety of contextual parameters which bear on the interpretation; for instance, sometimes the

interpretation of a textual source of a rule (and its associated formalisation) is associated with a jurisdiction, e.g. regional, national, or international levels, meaning that in one jurisdiction, the rule is interpreted one way, while in another jurisdication, it is intepreted in another way. To represent such parameters, we introduce the <lrml:Context> block, which permits the description of all the characteristics that are linked to a particular rule (e.g. rule1) using the operator <appliesXXX>, substituting the following relationships for XXX:

- Rule1 *has* TemporalCharateristics tblock1
- Rule1 *has* Strength defeasible
- Rule1 *has* Author aut1
- Rule1 *has* Jurisdiction US
- Rule1 *has* Authority Congress
- Rule1 *has* Source sec504-clsc-pnt1

For example, we have the following contextual parameters for rule1.

```
<lrml:Context key="ruleInfo1" hasCreationDate="#t8">
  <lrml:appliesTemporalCharacteristics
    keyref="#tblock1"/>
  <lrml:appliesStrength iri="&lrmlv;defeasible"/>
  <lrml:appliesRole>
    <lrml:Role iri="&lrmlv;#Author">
      <lrml:filledBy keyref="#aut1"/>
    </lrml:Role>
  </lrml:appliesRole>
  <lrml:appliesAuthority keyref="#congress"/>
  <lrml:appliesJurisdiction keyref="&jurisdictions;
    us"/>
  <lrml:appliesSource keyref="#sec504-clsc-pnt1"/>
  <lrml:toStatement keyref="#rule1"/>
</lrml:Context>
```

3. MODELLING NORMS

A key aspect of any attempt to represent, formalise, and reason with legal provisions is to provide a sound conceptual basis for such endeavors. The representation should be well grounded in the legal domain, allowing legal domain experts to understand the language in which norms are represented and reason with; at the same time, the language should properly formalised and amenable to computation. In addition, the representation should be transparent to the end users. Rules satisfy such requirements, where norms are represented by rules with the form [Sartor, 2005]:

$$\text{if } A_1, \ldots, A_n \text{ then } B$$

where $A_1, \ldots, A_n$ are the "pre-conditions of the norm", B is the "effect of the norm", and "if ... then ..." is a "normative conditional". One of the aims of the LegalRuleML specifications is to propose an XML representation for such rules.

In the following subsections, we present several components for modelling norms. First, we take it that normative conditionals are defeasible [Sartor, 2005], so we present our approach to defeasibility. Second, norms may also be modelled using deontic concepts, which themselves have associated notions of penalty and reparation. Finally, in contrast to normative conditionals and for use in the example, we have a representation of facts.

3.1 Defeasibility

The first use of defeasible rules is to capture conflicting rules/norms without making the resulting set of rules inconsistent. Given that -expression means the negation of expression, the following two rules conclude with the negation of each other

```
body_1 => head
body_2 => -head
```

Without defeasibile rules, rules with conclusions that are negations of each other could give rise to a contradiction, i.e., head and -head. Instead, defeasible reasoning is sceptical; that is, in case of a conflict such as the above, it refrains from taking any of the two conclusions, unless there are mechanisms to solve the conflict (see the discussion below on the superiority relation).

Notice that an application of this is to model exceptions. Exceptions limit the applicability of basic norms/rules, for example:

```
body => head
body, exception_condition => -head
```

In this case, the second rule is more specific than the first, and thus it forms an exception to the first, i.e., a case where the rule has extra conditions that encode the exception, blocking the conclusion of the first rule. Often, exceptions in defeasible reasoning can be simply encoded as

```
body => head
exception_condition => -head
```

In the definition of rules as normative conditionals made up of pre-conditions and effect, we can see a rule as a binary relationship between the pre-conditions (or body) of the rule, and the (legal) effect (head) of the rule. Formally, a rule can be defined by the following signature:

$$\text{body} \times \text{head}$$

We can then investigate the nature of such a relationship. Given two sets, we have the following seven possible relationships describing the "strength" of the connections between the body and the head of a rule:

```
body always head
body sometimes head
body not complement head
body no relationship head
body always complement head
body sometimes complement head
body not head
```

In defeasible logic we can represent the relationships using the following formalisation of rules (rule types):

```
body -> head
body => head
body ~> head
body -> -head
body => -head
body ~> -head
```

The seventh case is when there are no rules between the body and the head. The following table summarises the relationships, the notation used for them, and the strength of the relationship.[1]

Head/body relationship	Notation	Strength
body always head	body -> head	strict
body sometimes head	body => head	defeasible
body not complement head	body ~> head	defeater
body no relationship head		
body always complement head	body -> -head	strict
body sometimes complement head	body => -head	defeasible
body not head	body ~> -head	defeater

[1]The syntax presented here is based on Defeasible Logic, see Nute [1994], Antoniou et al. [2001].

The meaning of the different types of rules is as follows: For a *strict rule* `body -> head` the interpretation is that every time the body holds then the head holds.

For a *defeasible rule* `body => head` the reading is when the body holds, then, typically, the head holds. Alternatively we can say that the head holds when the body does unless there are reasons to assert that the head does not hold. This captures that it is possible to have exceptions to the rule/norm, and it is possible to have prescriptions for the opposite conclusion.

For a *defeaters* `body ~> head` the intuition is as follows: defeaters are rules that cannot establish that the head holds. Instead they can be use to specify that the opposite conclusion does not hold. In argumentation two types of defeaters are recognized: undercutting defeaters and rebutting defeaters. Undercutting defeaters are used in situations where there is an argument attacking the preconditions of a rule (or an argument), while the use of rebutting defeaters is to say that there is no relationship between the premises of an argument (preconditions of a rule or body) and the conclusion of the argument (effect of the rule or head).

Given the possibility to have conflicting rules, i.e., rules with opposite or contradictory heads, for example

```
body1 => head
body2 => -head
```

systems for defeasible reasoning include mechanisms to solve such conflicts. Different methods to solve conflicts have been proposed: *specificity*, *salience*, and *preference relation*. According to specificity, in case of a conflict between two rules, the most specific rule prevails over the less specific one, where a rule is more specific if its body subsumes the body of the other rule. For salience each rule has an attached salience or weight, where in case of a conflict between two rules, the one with the greatest salience or weight prevails over the other. Finally, a preference relation (also known as superiority relation) defines a binary relation over rules, where an element of the relation states the relative strength between two rules. Thus, in case of a conflict between two rules, if the preference relation is defined order such rules, the strongest of the two rules wins over the other.

Specificity corresponds to the well know legal principle of *lex specialis*. Prakken and Sartor [1997] argue that specificity is not always appropriate for legal reasoning and that there are other well understood legal principles such as *lex superior* and *lex posterior* apply instead. Prakken and Sartor [1997] cite cases in which the *lex specialis* principle might not the one used to solve the conflict, for example, a more specific article from a local council regulation might not override a less specific constitutional norm. Prakken and Sartor [1997] propose to use a dynamic preference relation to handle conflicting rules. The preference relation is dynamic in the sense that it is possible to argue about which instances of the relation hold and under which circumstances. Antoniou [2004] proposes that instances of the superiority relation appear in the head of rules, namely:

```
body => superiority
```

where `superiority` is a statement with the form

```
r1 > r2
```

where `r1` and `r2` are rule identifiers.

Gordon et al. [2007] propose Carneades as a rule based argumentation system suitable for legal reasoning where they use weights attached to the arguments (rules) to solve conflicts and to define proof standards. Governatori [2011] shows how to use the weights to generate an equivalent preference relation, and, consequently, how to capture the proposed proof standards. In addition Governatori [2011] shows that there are situations where a preference relation cannot be captured by using weights on the rules.

To handle defeasibility LegalRuleML has to capture the superiority relation and the strength of rules. For the superiority relation LegalRuleML offers the element `<Overrides>`, which defines a relationship of superiority where `cs2` overrides `cs1`. This can of elements are included in `hasQualification` blocks.

```
<lrml:hasQualification>
  <lrml:Overrides over="#cs1" under="#cs2"/>
</lrml:hasQualification>
```

For the representation of the strength of rules LegalRuleMLhas two options: The first is to include it in a `<Context>` block:

```
<lrml:Context key="ruleInfo2">
  <lrml:appliesStrength
    iri="&defeasible-ontology;#defeasible2"/>
  <lrml:toStatement keyref="#cs1"/>
</lrml:Context>
```

The second (and optional) way to express the qualification of the rule is directly inside of the rule, with an `hasStrength` block, namely:

```
<lrml:hasStrength>
  <lrml:Defeater key="str4"/>
</lrml:hasStrength>
```

3.2 Constitutive and Prescriptive Rules

In Legal Theory norms are classified mostly in two main categories: *constitutive norms* and *prescriptive norms*, which will be then represented as *constitutive rules* (also known as *counts-as rules*) and *prescriptive rules*.[2]

The function of constitutive norms is to define and create the so called *institutional facts* [Searle, 1996]. Where an institutional fact is how a particular concept is understood in a specific institution. Thus, constitutive rules provide definitions of the terms and concept used in a jurisdiction. On the other hand the scope of prescriptive rules is to dictate what are the obligations, prohibitions, permissions ... in a legal system, and the conditions under which we have them. LegalRuleML uses deontic operators to capture such notions. Deontic operators are meant to qualify formulas. A Deontic operator takes as its argument a formula and returns a formula. For example, given the (atomic) formula `PayInvoice(guido)`, meaning 'Guido pays the invoice', and the deontic operator `[OBL]` (for obligation), the application of the deontic operator to the formula generates the new formula `[OBL]PayInvoice(guido)`, meaning that "it is obligatory that Guido pays the invoice".

The deontic operators currently defined in the core of LegalRuleML are given in the the following XSD fragment:

```
<xs:group name="Deontic.Node.choice">
  <xs:choice>
    <xs:group ref="lrml:Obligation.Node.choice"/>
    <xs:group ref="lrml:Permission.Node.choice"/>
    <xs:group ref="lrml:Prohibition.Node.choice"/>
    <xs:group ref="lrml:Right.Node.choice"/>
    <xs:group ref="lrml:Violation.Node.choice"/>
    <xs:group ref="lrml:Compliance.Node.choice"/>
  </xs:choice>
</xs:group>
```

[2]Gordon et al. [2009] identify more types of norms/rules. However, most of them can be reduced to the two types described here insofar as the distinction is not on structure of the rules but it depends on the meaning of the content (specific effect) of the rules, while keeping the same logical format.

The operators Obligation, Permission and Prohibition are the standard operators of deontic logic. The deontic operators admit children nodes for specifying the Bearer and the AuxiliaryParty of the operator, for example of the operator specifying a right we have:

```
<lrml:Right>
  <ruleml:slot>
    <lrml:Bearer
      iri="&deontic-ontology;#oblbsub1"/>
    <ruleml:Ind>X</ruleml:Ind>
  </ruleml:slot>
  <ruleml:slot>
    <lrml:AuxiliaryParty
      iri="&deontic-ontology;#oblbAdd1"/>
    <ruleml:Ind>Y</ruleml:Ind>
  </ruleml:slot>
  <ruleml:Atom>
    <ruleml:Rel iri="#copyright"/>
    <ruleml:Var>X</ruleml:Var>
    <ruleml:Ind>book</ruleml:Ind>
  </ruleml:Atom>
</lrml:Right>
```

Thus LegalRuleML can model the well understood semantics of Right which is a permission on one party implying an obligation (or prohibition) on a second party.

Violation and Compliance have a nature different from the others, and instead of taking a formula as their argument, the argument is a reference to a rule. Thus the meaning is that the rule the Violation operator refers to has been violated, similarly for Compliance, the rule has been complied with. Given the intended reading these two operators can only appear in the body of rules.

Constitutive rules have the following form:

```
body => head
```

Where body is a set of formulas and deontic formulas, and head is a formula. Thus constitutive rules do not allow deontic formulas in the head part or consequent of the rule. This restriction is due to the meaning of this type of rule which is intended to define concepts and not to prescribe behaviours.

```
<lrml:ConstitutiveStatement key="cs1">
  <ruleml:Rule key=":key1">
    <lrml:hasStrength>
      strength of the rule
    </lrml:hasStrength>
    <ruleml:if>
      set of formulas and deontic formulas
    </ruleml:if>
    <ruleml:then>
      formula
    </ruleml:then>
  </ruleml:Rule>
</lrml:ConstitutiveStatement>
```

The if part corresponds to the body of the rule and the then part is the head of the rule. LegalRuleML defines specifics a sophisticated mechanism of prescriptiveness to prevent them to occur in the head of constitutive rules. To implement this constraint RuleML was extended and integrated with LegalRuleML.

For prescriptive rules the form is as follows:

body => $[D_1]$formula$_1$,...,$[D_n]$formula$_n$

where body is a set of formulas and deontic formulas and the head is a list of deontic formulas (where $[D_i]$ are deontic operators). The following is the LegalRuleML format for prescriptive rules.

```
<lrml:PrescriptiveStatement key="ps1">
  <ruleml:Rule key=":key1">
    <lrml:hasStrength>
      strength of the rule
    </lrml:hasStrength>
    <ruleml:if>
      set of deontic formulas and formulas
    </ruleml:if>
    <ruleml:then>
      <lrml:SuborderList>
        list of deontic formulas
      </lrml:SuborderList>
    </ruleml:then>
  </ruleml:Rule>
</lrml:PrescriptiveStatement>
```

The difference between constitutive rules and prescriptive rules is in the content of the head where the head of a prescriptive rule is list of deontic operators, i.e., $[D_1]$formula$_1$,...,$[D_n]$formula$_n$ which is called *suborder* list, and represented in LegalRuleML by the <lrml:Suborder> block.

3.3 Suborder, Penalty and Reparation

One of the characteristics of the norms is the possibility to violate them and to compensate their violation penalties. To model this feature of norms and legal reasoning Governatori and Rotolo [2006] introduced what is called here suborder list, and Governatori [2005] show how to combine them with defeasible reasoning for the modelling of (business) contracts.

As we have seen above a suborder list is a list of deontic formulas, i.e., formulas of the form [D]A, where [D] is one of [OBL] (obligation), [FORB] (prohibition, or forbidden), PERM (permission) and [RIGHT] (right). Suborder list have the restriction that if $[D_i]$ is either [PERM] or [RIGHT] the elements after it cannot be [OBL] or [FORB].

To illustrate the meaning of suborder lists, consider the following example

```
[OBL]A, [OBL]B, [FORB]C, [PERM]D
```

The expression means that A is obligatory, but if it is violated, i.e., we have its opposite -A, then the obligation in force to compensate the violation of [OBL]A is B. If also this obligation of B is violated, then we have the prohibition of C. At this stage, if we have a violation of such a prohibition, i.e., we have C, then the permission of D kicks in.

Governatori and Rotolo [2006], Governatori [2005] also discuss mechanisms to combine the suborder lists from different rules. For example, given the rules

```
body => [OBL]A
-A => [OBL]B
```

Here the body of the second rule is the negation of the content of the obligation in the head of the first rule. It is possible to merge the two rules above in the following rule

```
body => [OBL]A, [OBL]B
```

stating that to compensate the violation of the obligation of A one as the obligation of B. This suggests that suborder lists provide a simple and convenient mechanism to model penalties. It is not uncommon for legal text (for example contracts) to include sections

about penalties (see Section 4 below for a concrete example of this phenomenon), and one penalty can be related as compensation for many norms. To model this and to maintain the isomorphism between a sources and its formalisation LegalRuleML includes a <Penalty> element whose scope is to represent a penalty as a suborder list (including the trivial not empty list of a single element).

```
<lrml:Penalty key="pen2">
  <lrml:SuborderList>
    list of deontic formulas
  </lrml:SuborderList>
</lrml:Penalty>
```

LegalRuleML not only models penalty, but aims to connect the penalty with the correspondent reparation:

```
<lrml:Reparation key="rep1">
  <lrml:appliesAssociation>
    <lrml:Association key="assoc1">
      <lrml:appliesPenalty keyref="#pen1"/>
      <lrml:toTarget keyref="#ps1"/>
    </lrml:Association>
  </lrml:appliesAssociation>
</lrml:Reparation>
```

With the temporal model of LegalRuleML, we can model a unique deontic rule (e.g. prohibition) and several penalties that are updated over the time according to the modifications of the law. Dynamically the legal reasoner can point out to the correct penalty according to the time of the crime. (e.g. statutory damage 500$ in 2000, 750$ in 2006, 1000$ in 2010).

The last feature of LegalRuleML we discuss here is that it has the ability to include factual statements using `FactualStatement` blocks where the content of the statements is modelled with simple RuleML <Atom> elements.

```
<lrml:FactualStatement key="fact1">
  <ruleml:Atom key=":atom11">
    <ruleml:Rel iri="#rel5"/>
    <ruleml:Ind iri="#JohnDoe"/>
  </ruleml:Atom>
</lrml:FactualStatement>
```

4. A COMPREHENSIVE EXAMPLE

In this section we illustrate the use of LegalRuleML by modelling a fragment of Section 29 of the Australian "National Consumer Credit Protection Act 2009" (Act No. 134 of 2009). The section of the act entitled "Prohibition on engaging in credit activities without a licence" recites

> (1) A person must not engage in a credit activity if the person does not hold a licence authorising the person to engage in the credit activity.
> Civil penalty: 2,000 penalty units.
> [. . .]
> Criminal penalty: 200 penalty units, or 2 years imprisonment, or both.

In the norm above we can notice that the penalties are stated as separate statements. Accordingly the best way to capture the structure is to use <Penalty> elements for them. The text of the provision can be paraphrased as follows:

- It is forbidden for a person to engage in a credit activity.
- A person is permitted to engage in a credit activity if the person hold a licence

Based on the observation and paraphrases above we can model the norm with the following rules (and auxiliary statements)

```
ps1: Person(x) => [FORB]EngageCreditActivity(x)
ps2: HasLicence(x) => [PERM]EngageCreditActivity(x)
ps2 > ps1
pen1: [OBL] PayCivilUnits(x,2000)
pen2: [OBL] PayPenalUnits(x,200),
      [OBL] Imprisonment(x,2m),
      [OBL] PayPenaltyUnitsAndImprisonment(x,200,2m)
rep1: [Violation]ps1, pen1
rep2: [Vioaltion]ps1, pen2
```

This norm can be represented in LegalRuleML as follows:

```
<lrml:LegalSources>
  <lrml:LegalSource key="ls1"
    sameAs="http://www.comlaw.gov.au/Details/
      C2009A00134/Html/Text#param43"/>
</lrml:LegalSources>
```

the block above is for declaring the source of the legal provisions and to give it a key to refer to it. After that we an `Associations` block allows to link legal provisions with the rules (and other statements) modelling them

```
<lrml:Context key="psInfo1">
  <lrml:appliesAssociations>
    <lrml:Associations>
      <lrml:Association>
        <lrml:appliesSource keyref="#ls1"/>
        <lrml:toTarget keyref="#ps1"/>
        <lrml:toTarget keyref="#ps2"/>
        <lrml:toTarget keyref="#pen1"/>
        <lrml:toTarget keyref="#pen2"/>
      </lrml:Association>
    </lrml:Associations>
  </lrml:appliesAssociations>
</lrml:Context>
```

In this case we have that the norm referred to the key `ls1` is modelled by a set of statements, namely `ps1`, `ps2`, `pen1`, `pen2`. The LegalRuleML statements for representing the norms are given in the code below.

```
<lrml:Statements key="textblock1">
  <lrml:hasQualification>
    <lrml:Overrides over="#ps2" under="#ps1"/>
  </lrml:hasQualification>
  <lrml:PrescriptiveStatement key="ps1">
    <ruleml:Rule key=":rule1" closure="universal">
      <lrml:hasStrength>
        <lrml:Defeasible/>
      </lrml:hasStrength>
      <ruleml:if>
        <ruleml:Atom>
          <ruleml:Rel iri="#person"/>
          <ruleml:Var>X</ruleml:Var>
        </ruleml:Atom>
      </ruleml:if>
      <ruleml:then>
        <lrml:SuborderList>
          <lrml:Prohibition>
            <ruleml:Atom>
              <ruleml:Rel
                iri="#engageCreditActivity"/>
```

```
            <ruleml:Var>X</ruleml:Var>
          </ruleml:Atom>
        </lrml:Prohibition>
      </lrml:SuborderList>
    </ruleml:then>
  </ruleml:Rule>
</lrml:PrescriptiveStatement>
<lrml:PrescriptiveStatement key="ps2">
  <ruleml:Rule key=":rule2" closure="universal">
    <lrml:hasStrength>
      <lrml:Defeasible/>
    </lrml:hasStrength>
    <ruleml:if>
      <ruleml:Atom>
        <ruleml:Rel iri="#hasLicence"/>
        <ruleml:Var>X</ruleml:Var>
      </ruleml:Atom>
    </ruleml:if>
    <ruleml:then>
      <lrml:SuborderList>
        <lrml:Permission>
          <ruleml:Atom>
            <ruleml:Rel
              iri="#engageCreditActivity"/>
            <ruleml:Var>X</ruleml:Var>
          </ruleml:Atom>
        </lrml:Permission>
      </lrml:SuborderList>
    </ruleml:then>
  </ruleml:Rule>
</lrml:PrescriptiveStatement>
<lrml:Penalty key="pen1">
  <lrml:SuborderList>
    <lrml:Obligation>
      <ruleml:Atom>
        <ruleml:Rel iri="#payCivilUnits"/>
        <ruleml:Var>X</ruleml:Var>
        <ruleml:Ind>2000</ruleml:Ind>
      </ruleml:Atom>
    </lrml:Obligation>
  </lrml:SuborderList>
</lrml:Penalty>
<lrml:Penalty key="pen2">
  <lrml:SuborderList>
    <lrml:Obligation>
      <ruleml:Atom>
        <ruleml:Rel iri="#payPenalUnits"/>
        <ruleml:Var>X</ruleml:Var>
        <ruleml:Ind>200</ruleml:Ind>
      </ruleml:Atom>
    </lrml:Obligation>
    <lrml:Obligation>
      <ruleml:Atom>
        <ruleml:Rel iri="#imprisonment"/>
        <ruleml:Var>X</ruleml:Var>
        <ruleml:Ind>2 months</ruleml:Ind>
      </ruleml:Atom>
    </lrml:Obligation>
    <lrml:Obligation>
      <ruleml:Atom>
        <ruleml:Rel
          iri="#payPenalUnitAndImprisonment"/>
        <ruleml:Var>X</ruleml:Var>
        <ruleml:Ind>200</ruleml:Ind>
        <ruleml:Ind>2 months</ruleml:Ind>
      </ruleml:Atom>
    </lrml:Obligation>
  </lrml:SuborderList>
</lrml:Penalty>
<lrml:Reparation key="rep1">
  <lrml:appliesAssociation>
    <lrml:Association key="assoc1">
      <lrml:appliesPenalty keyref="#pen1"/>
      <lrml:toTarget keyref="#ps1"/>
    </lrml:Association>
  </lrml:appliesAssociation>
</lrml:Reparation>
<lrml:Reparation key="rep2">
  <lrml:appliesAssociation>
    <lrml:Association keyref="assoc1">
      <lrml:appliesPenalty keyref="#pen2"/>
      <lrml:toTarget keyref="#ps1"/>
    </lrml:Association>
  </lrml:appliesAssociation>
</lrml:Reparation>
</lrml:Statements>
```

5. RELATED WORK

In this section, we review related work.

5.1 LKIF

Legal Knowledge Interchange Format (LKIF) is a specification that includes a legal core ontology and a legal rule language that can be used to deploy comprehensive legal knowledge management solutions. As LKIF is open source, it can be used without becoming dependent on proprietary products of particular vendors [Estrella Project, 2008, Hoekstra et al., 2009, Gordon, 2008]. It was developed in the ESTRELLA project (ESTRELLA IST-2004-027655) and used in Carneadeas tool [Gordon et al., 2007]. LegalRuleML takes inspiration from LKIF, particularly in terms of closely representing legal knowledge and legal reasoning. Despite its power, LKIF lacks aspects that appear in LegalRuleML: temporal aspects are not linkable to any part of a rule with arbitrary granularity; the temporal aspect is not managed with appropriate operators; violation-reparation and, in general, behaviors are not richly modelled; defeasibility is not linked to the temporal parameters; suborder lists of atoms, where the order is determinant, are not possible.

5.2 RuleML

RuleML is a family of languages, whose modular system of schemas for XML permits high-precision web rule interchange. The family's top-level distinction is deliberation rules vs. reaction rules. Deliberation rules include modal and derivation rules, which themselves include facts, queries (incl. integrity constraints), and Horn rules (incl. Datalog). Reaction rules include Complex Event Processing (CEP), Knowledge Representation (KR), Event-Condition-Action (ECA) rules, as well as Production (CA) rules. RuleML rules can combine all parts of both derivation and reaction rules. This allows uniform XML serialization across all kinds of rules. After its use in SWRL and SWSL, RuleML has provided strong input to W3C RIF on several levels. This includes the use of 'striped' XML as well as the structuring of rule classes into sublanguages with partial mappings between, e.g., Datalog RuleML and RIF-Core, Hornlog RuleML and RIF-BLD, as well as Production RuleML and RIF-PRD. After preparatory work on Policy RuleML (`http://policy.ruleml.org`), the recent LegalRuleML speci-

fication described in this paper constitutes a major collaboration between OASIS and RuleML.

5.3 RIF

RIF (http://www.w3.org/2005/rules/wiki/RIF_Working_Group) is a W3C recommendation, influenced by RuleML, to define a standard Web Rule Interchange Format for facilitating the exchange of rule sets among different rule systems and to facilitate the development of intelligent rule-based applications for the Semantic Web. RIF is a general Web rule language in that makes the use of IRIs (Internationalized Resource Identifiers) mandatory and supports XML Schema data types. The RIF architecture is conceived as a family of languages, called dialects. A RIF dialect is a rule-based language with an XML syntax and a well-defined semantics. So far, the RIF working group has defined the Basic Logic Dialect (RIF-BLD), which semantically corresponds to a definite Horn rule language with equality and some syntactic extensions such as F-logic-like frames, and the Production Rule Dialect (RIF-PRD) for representing classical production rules. The intersection condition language of RIF-BLD and RIF-PRD is defined in the RIF-Core. For defining further logic-based dialects the RIF Working Group has also specified the Framework for Logic Dialects (RIF-FLD). RIF-FLD uses a uniform notion of terms for both expressions and atoms in a Hilog-like manner. The connection to other W3C Semantic Web languages in Semantic Web stack is established via Semantic Web Compatibility (RIF-SWC). RIF does not provide direct support for adequate representation of legal rules and legal reasoning. The current RIF dialects are not expressive enough, since they do not support e.g. logic-based negation, non-monotonic reasoning, events and temporal metadata etc.

5.4 SBVR

The Semantics of Business Vocabulary and Business Rules (SBVR) provides a controlled natural language [Wyner et al., 2010] of fixed and finite vocabulary and syntactic form for the expression of the terminology, facts, and rules for business documents across a range of business activities and organisations (http://www.omg.org/spec/SBVR/1.0/). Such documents represent a natural and accessible way to express the conceptual structure and operational controls of a business, yet, as a CNL, the business rules can be represented in predicate logic (with some modal logic), systematically managed, and converted to machine-executable form. In addition, SBVR has an associated XML Metadata Interchange (XMI), which supports the interchange of documents across businesses. Thus, an organisation can form an appropriate sublanguage that is comprehensible, processable, and interchangeable with other organisations using SBVR. Without a CNL, logical representation, management, execution, and interchange would be problematic given the current state of natural language processing.

SBVR and LegalRuleML are closely related in that both provide XML encodings of the semantics of terminology, facts, and rules, although these encodings are not aligned. SBVR bears on business rules, which may or may not have legal standing; LegalRuleML represents statements of legal standing, particularly legal concepts and processes (e.g. *authority* and *jurisdiction*). The temporal notions of *enforceability*, *efficacy*, and *applicability* are not provided in SBVR. LegalRuleML enables the expression of *defeasibility*, a rich range of *deontic concepts*, and associated concepts of *penalty* and *reparation*. Finally, SBVR is first a CNL and is not related to RuleML languages, while LegalRuleML has no associated CNL, but relates to RuleML languages.

6. CONCLUSIONS AND FUTURE WORK

To facilitate the creation of machine-readable forms of the contents of legal texts, the OASIS TC on LegalRuleML created the first version of LegalRuleML. This version is intended to address three general needs: to provide an XML that satisfies the requirements of the legal domain, to relate textual provisions with the markup language, and to use the Linked Open Data approach. To address these needs, a range of requirements and design principles are specified. Key elements of LegalRuleML that satisfy these requirements and principles are discussed and illustrated, covering sources, time, defeasibility, and reparation, among several other elements. A comprehensive example is provided. LegalRuleML is compared with related languages, noting what is adopted or adapted.

In future work, the TC will develop tools to translate between source text and LegalRuleML, provide further examples, and extend the language to address meta-rules, complex event modelling, case-law modelling, argumentation, procedural representations, and other features needed to represent legal texts.

7. ACKNOWLEDGEMENTS

This work is part of the effort of the OASIS LegalRuleML Technical Committee dedicated to the development of a standard for the representation and exchange of legal knowledge.

NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

References

G. Antoniou. Defeasible logic with dynamic priorities. *International Journal of Intelligent Systems*, 19(5):463–472, 2004.

G. Antoniou, D. Billington, G. Governatori, and M. J. Maher. Representation results for defeasible logic. *ACM Transactions on Computational Logic*, 2(2):255–287, 2001.

K. Ashley and T. M. van Engers, editors. *Proceedings of the 13th International Conference on Artificial Intelligence and Law*, Pittsburgh, PA, USA, June 6-10 2011. ACM Press.

T. Bench-Capon and F. P. Coenen. Isomorphism and legal knowledge based systems. *Artificial Intelligence and Law*, 1(1):65–86, 1992.

V. R. Benjamins, P. Casanovas, J. Breuker, and A. Gangemi, editors. *Law and the Semantic Web: Legal Ontologies, Methodologies, Legal Information Retrieval and Applications*. Springer, 2005.

T. Berners-Lee. Long live the web: A call for continued open standards and neutrality. *Scientific American*, (22 November): 1–6, 2010.

A. Boer, R. Winkels, and F. Vitali. Metalex XML and the legal knowledge interchange format. In P. Casanovas, G. Sartor, N. Casellas, and R. Rubino, editors, *Computable Models of the Law, Languages, Dialogues, Games, Ontologies*, volume 4884 of *Lecture Notes in Computer Science*, pages 21–41. Springer, 2008.

J. Breuker, A. Boer, R. Hoekstra, and K. van den Berg. Developing content for lkif: Ontologies and frameworks for legal reasoning. In T. M. van Engers, editor, *JURIX*, volume 152 of *Frontiers in Artificial Intelligence and Applications*, pages 169–174. IOS Press, 2006.

Estrella Project. The legal knowledge interchange format (LKIF). Technical report, ESTRELLA Project, 2008. URL http://www.estrellaproject.org/doc/Estrella-D4.1.pdf.

E. Francesconi, S. Montemagni, W. Peters, and D. Tiscornia, editors. *Semantic Processing of Legal Texts: Where the Language of Law*

Meets the Law of Language, volume 6036 of *Lecture Notes in Computer Science*, 2010. Springer.

T. Gordon, H. Prakken, and D. Walton. The Carneades model of argument and burden of proof. *Artificial Intelligence*, 171: 875–896, 2007.

T. F. Gordon. *The Pleadings Game; An Artificial Intelligence Model of Procedural Justice*. Springer, New York, 1995.

T. F. Gordon. Constructing legal arguments with rules in the legal knowledge interchange format (LKIF). In P. Casanovas, N. Casellas, R. Rubino, and G. Sartor, editors, *Computable Models of the Law*, number 4884 in Lecture Notes in Computer Science, pages 162–184. Springer Verlag, 2008.

T. F. Gordon, G. Governatori, and A. Rotolo. Rules and norms: Requirements for rule interchange languages in the legal domain. In Governatori et al. [2009], pages 282–296.

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

G. Governatori. On the relationship between Carneades and defeasible logic. In T. van Engers, editor, *Procedings of the 13th International Conference on Artificial Intelligence and Law (ICAIL 2011)*. ACM Press, 2011.

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

G. Governatori and A. Rotolo. Norm compliance in business process modeling. In M. Dean, J. Hall, A. Rotolo, and S. Tabet, editors, *RuleML*, volume 6403 of *Lecture Notes in Computer Science*, pages 194–209. Springer, 2010.

G. Governatori, A. Rotolo, and G. Sartor. Temporalised normative positions in defeasible logic. In *Procedings of the 10th International Conference on Artificial Intelligence and Law (ICAIL 2005)*, pages 25–34. ACM, 2005.

G. Governatori, J. Hall, and A. Paschke, editors. *Rule Interchange and Applications, International Symposium, RuleML 2009, Las Vegas, Nevada, USA, November 5-7, 2009. Proceedings*, volume 5858 of *Lecture Notes in Computer Science*, 2009. Springer.

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

R. Hoekstra, J. Breuker, M. D. Bello, and A. Boer. LKIF core: Principled ontology development for the legal domain. In *Law, Ontologies and the Semantic Web*, pages 21–52, Amsterdam, The Netherlands, 2009.

H. Kelsen. *Reine Rechtslehre*. Wien, 2nd edition, 1960.

C. Lupo, F. Vitali, E. Francesconi, M. Palmirani, R. Winkels, E. de Maat, A. Boer, and P. Mascellani. General XML format(s) for legal sources - Estrella European Project IST-2004-027655. Technical report, Faculty of Law, University of Amsterdam, Amsterdam, The Netherlands, 2007.

D. Nute. *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 3, chapter Defeasible Logic, pages 353–395. Oxford University Press, Oxford, 1994.

M. Palmirani, G. Contissa, and R. Rubino. Fill the gap in the legal knowledge modelling. In Governatori et al. [2009], pages 305–314.

M. Palmirani, G. Governatori, and G. Contissa. Temporal dimensions in rules modelling. In R. Winkels, editor, *JURIX*, volume 223 of *Frontiers in Artificial Intelligence and Applications*, pages 159–162. IOS Press, 2010.

H. Prakken and G. Sartor. A dialectical model of assessing conflicting argument in legal reasoning. *Artificial Intelligence and Law*, 4(3-4):331–368, 1996.

H. Prakken and G. Sartor. Argument-based extended logic programming with defeasible priorities. *Journal of Applied Non-Classical Logics*, 7(1), 1997.

G. Sartor. Legal reasoning: A cognitive approach to the law. In E. Pattaro, H. Rottleuthner, R. Shiner, A. Peczenik, and G. Sartor, editors, *A Treatise of Legal Philosophy and General Jurisprudence*, volume 5, page 844. Springer, 2005.

G. Sartor. Legal concepts as inferential nodes and ontological categories. *Artificial Intelligence and Law*, 17(3):217–251, 2009.

J. R. Searle. *The Construction of Social Reality*. The Free Press, New York, 1996.

A. Wyner, K. Angelov, G. Barzdins, D. Damljanovic, B. Davis, N. Fuchs, S. Hoefler, K. Jones, K. Kaljurand, T. Kuhn, M. Luts, J. Pool, M. Rosner, R. Schwitter, and J. Sowa. On controlled natural languages: properties and prospects. In *Proceedings of the 2009 conference on Controlled natural language*, CNL'09, pages 281–289, Berlin, Heidelberg, 2010. Springer-Verlag.