Rules and Norms: Requirements for Rule Interchange Languages in the Legal Domain

Thomas F. Gordon[1], Guido Governatori[2], and Antonino Rotolo[3]

[1] Fraunhofer FOKUS, Berlin, Germany
thomas.gordon@fokus.fraunhofer.de
[2] NICTA, Queensland Research Laboratory, Brisbane, Australia
guido.governatori@nicta.com.au
[3] CIRSFID, University of Bologna, Bologna, Italy
antonino.rotolo@unibo.it

Abstract. In this survey paper we summarize the requirements for rule interchange languages for applications in the legal domain and use these requirements to evaluate RuleML, SBVR, SWRL and RIF. We also present the Legal Knowledge Interchange Format (LKIF), a new rule interchange format developed specifically for applications in the legal domain.

1 Introduction

An extensive research has been devoted in the last years for developing rule languages in the legal domain. Interesting efforts has been carried out especially in the field of e-contracting, business processes and automated negotiation systems. This led to devise new languages, or adjust existing ones, specifically designed for documenting and modeling the semantics of business vocabularies, facts, and rules. Significant examples are SBVR [39], the case handling paradigm [43], OWL-S [29], ContractLog [30], Sadiq et al.'s constraint specification framework [37], the Web Service Modeling Ontology (WSMO) [33], the ConDec language [31], PENELOPE [14], and RuleML for business rules [21, 16].

But legal rules are not only pervasive in modeling e-transactions—where formalizing and handling legal rules and contract clauses is required for providing tools to support legally valid interactions and/or to legally ground contractual transactions—but their sound and faithful representation is obviously crucial for representing legislative documents, regulations, and other sources of law (for instance, in the domains of e-governance and e-government).

Since the seminal work of Sergot et al. [41], which formalized the British Nationality Act in a logic programming setting, the AI & Law community has devoted an extensive effort for modeling many aspects of legal rules and regulations[4]. However,

[4] The interested reader may consult a large number of relevant works published in the Artificial Intelligence and Law journal and in the proceedings of conferences such as ICAIL and JURIX.

there are still a few works which address the problem of devising rule interchange languages, properly speaking, for the legal domain. LKIF is probably the first systematic attempt in this regard (LKIF will be discussed here in Section 3.5).

Significant experiences for representing legislative documents throughout XML languages, for instance, are CEN MetaLex [7], SDU BWB (see [27]), LexDania (see [27]), NormeinRete [26], AKOMA NTOSO [2]. Other XML standards in legal domain are CHLexML [10], EnAct [3], Legal RDF [28], eLaw (see [27]), Legal XML (see [27]), LAMS [24], JSMS (see [27]), and UKMF (see [27]). Much of these XML-based attempts are ambitious, valuable and effective. Yet, they mostly focused on representing *legal documents* rather than modeling directly *legal rules*. In addition, some are focused on specific application areas, others model a few aspects of the many concerns that exist in reality, or, if developed in order to be sufficiently general, exhibit some limitations since they are not based on robust or comprehensive conceptual models for representing legal rules to be applied in the legal domain (for a detailed evaluation of modeling, e.g., legislation, see [27]).

In general, many of the drawbacks affecting many existing languages are perhaps due to the fact that there has not yet been an overall and systematic effort to establish a general list of requirements for rule interchange languages in the legal domain or because there is not yet an agreement in particular among the practitioners working in this field. This survey paper is meant to offer a list of minimal requirements to a large audience of computer scientists, legal engineers and practitioners who are willing to model legal rules. These requirements are then discussed with regard to some existing rule interchange languages.

Note that a remarkable, and additional difficulty is that it is sometimes not immediate to adjust and extend existing standards for rule interchange languages when we need to use them in the legal domain. Indeed, although the legal domain has several features which are shared by other domains, some aspects are very specific for the law. Consider the notion of information retrieval. In the law, question-answering "seems more relevant than information retrieval", since "question requires some deduction or inference before an appropriate answer can be given" and "regulations may contain many different articles about the same topic and one can only assess whether something is permitted or not by understanding the full documentation". "A rather detailed understanding is required, in particular, because regulations generally contain complex structures of exceptions" [6, p. 9]. The peculiarity of the legal domain thus poses specific problems for developing suitable and faithful representation languages.

The layout of the paper is as follows. In Section 2 we provide a rather comprehensive list of requirements for devising rule interchange languages. The subsequent sections discuss these requirements to evaluate RuleML, SBVR, SWRL, RIF, and LKIF. Some brief conclusions end the paper.

2 Requirements

The law is a complex phenomenon, which can be analyzed into different branches according to the authority who produces legal norms and according to the circumstances and procedures under which norms are created. But, independently of these aspects, it is possible to identify some general features that norms should enjoy.

First of all, it is widely acknowledged in legal theory and AI & Law that norms have basically a conditional structure like [23, 38]

$$\text{if } A_1, \ldots, A_n \text{ then } B \tag{1}$$

where $A_1, \ldots, A_n$ are the applicability conditions of the norm and B denotes the legal effect which ought to follow when those applicability conditions hold[5].

This very general view highlights an immediate link between the concepts of norm and rule. However, there are many types of rules. The common sense, dictionary meaning of rule is "One of a set of explicit or understood regulations or principles governing conduct within a particular sphere of activity." [1]. In classical logic, rules can be inference rules or material implications. In computer science, rules can be production rules, grammar rules, or rewrite rules.

When we use the term 'rule' in the legal field, we usually mean rule in the regulatory sense. But rules express not only regulations about how to act. For example, von Wright [45] classified norms into the following main types (among others):

1. determinative rules, which define concepts or constitute activities that cannot exist without such rules. These rules are also called in the literature 'constitutive rules'.
2. technical rules, which state that something has to be done in order for something else to be attained;
3. prescriptions, which regulate actions by making them obligatory, permitted, or prohibited. These norms, to be complete, should indicate
   - who (the norm-subjects)
   - does what (the action-theme)
   - in what circumstances (the condition of application) and
   - the nature of their guidance (the mode).

Notice that the notion of norm proposed by von Wright is very general and extends well over the notion of norm in legal reasoning; but in some cases the component of a rule have to modified. For example, legal systems can have provisions to handle changes in the systems itself. Thus, it is possible to have norms about how to change other norms. These rules have again a prescriptive character, but we have to adjust the element, in particular these rules should specify, *what* (the content to be modified), *how* (the new content), in what circumstances, and the nature of the modifications (e.g.,, substitution, derogation, abrogation, annulment, ...)

Many of these aspects have been acknowledged in the field of artificial intelligence and law, where there is now much agreement about the structure and properties of rules [15, 32, 22, 44, 38]. Important requirements for legal rule languages from the field of AI & Law include the following:

[5] Indeed, norms can be also unconditioned, that is their effects may not depend upon any antecedent condition. Consider, for example, the norm "everyone has the right to express his or her opinion". Usually, however, norms are conditioned. In addition, unconditioned norms can formally be reconstructed in terms of (1) with no antecedent conditions.

Isomorphism [5]. To ease validation and maintenance, there should be a one-to-one correspondence between the rules in the formal model and the units of natural language text which express the rules in the original legal sources, such as sections of legislation. This entails, for example, that a general rule and separately stated exceptions, in different sections of a statute, should not be converged into a single rule in the formal model.

Reification [15]. Rules are objects with properties, such as

- **Jurisdiction.** The limits within which the rule is authoritative and its effects are binding (of particular importance are spatial and geographical references to model jurisdiction).
- **Authority [32].** Who produced the rule, a feature which indicates the ranking status of the rule within the sources of law (whether the rule is a constitutional provision, a statute, is part of a contract clause or is the ruling of a precedent, and so on).
- **Temporal properties [19].** Rules usually are qualified by temporal properties, such as:
  1. the time when the norm is in force and/or has been enacted;
  2. the time when the norm can produce legal effects;
  3. the time when the normative effects hold.

Rule semantics. Any language for modeling legal rules should be based on a precise and rigorous semantics, which allows for correctly computing the legal effects that should follow from a set of legal rules.

Defeasibility [15, 32, 38]. When the antecedent of a rule is satisfied by the facts of a case, the conclusion of the rule presumably holds, but is not necessarily true. The defeasibility of legal rules breaks down into the following issues:

- **Conflicts [32].** Rules can conflict, namely, they may lead to incompatible legal effects. Conceptually, conflicts can be of different types, according to whether two conflicting rules
  - are such that one is an exception of the other (i.e., one is more specific than the other);
  - have a different ranking status;
  - have been enacted at different times;

  Accordingly, rule conflicts can be resolved using principles about rule priorities, such as:
  - *lex specialis*, which gives priority to the more specific rules (the exceptions);
  - *lex superior*, which gives priority to the rule from the higher authority (see 'Authority' above);
  - *lex posterior*, which gives priority to the rule enacted later (see 'Temporal parameters' above).
- **Exclusionary rules [32, 38, 15].** Some rules provide one way to explicitly undercut other rules, namely, to make them inapplicable.

Contraposition [32]. Rules do not counterpose. If some conclusion of a rule is not true, the rule does not sanction any inferences about the truth of its premises.

Contributory reasons or factors [38]. It is not always possible to formulate precise rules, even defeasible ones, for aggregating the factors relevant for resolving a legal issue. For example: "The educational value of a work needs to be taken into consideration when evaluating whether the work is covered by the copyright doctrine of fair use."

Rule validity [19]. Rules can be invalid or become invalid. Deleting invalid rules is not an option when it is necessary to reason retroactively with rules which were valid at various times over a course of events. For instance:

1. The *annulment* of a norm is usually seen as a kind of repeal which invalidates the norm and removes it from the legal system as if it had never been enacted. The effect of an annulment applies *ex tunc*: annulreled norms are prevented from producing any legal effects, also for past events.
2. An *abrogation* on the other hand operates *ex nunc*: The rule continues to apply for events which occured before the rule was abrogated.

Legal procedures. Rules not only regulate the procedures for resolving legal conflicts (see above), but also for arguing or reasoning about whether or not some action or state complies with other, substantive rules [16]. In particular, rules are required for procedures which

1. regulate methods for detecting violations of the law;
2. determine the normative effects triggered by norm violations, such as reparative obligations, namely, which are meant to repair or compensate violations[6].

Normative effects. There are many normative effects that follow from applying rules, such as obligations, permissions, prohibitions and also more articulated effects such as those introduced, e.g., by Hohfeld (see [38]). Below is a rather comprehensive list of normative effects [35]:

Evaluative, which indicate that something is good or bad, is a value to be optimised or an evil to be minimised. For example, "Human dignity is valuable", "Participation ought to be promoted";

Qualificatory, which ascribe a legal quality to a person or an object. For example, "*x* is a citizen";

Definitional, which specify the meaning of a term. For example, "Tolling agreement means any agreement to put a specified amount of raw material per period through a particular processing facility";

Deontic, which, typically, impose the obligation or confer the permission to do a certain action. For example, "*x* has the obligation to do *A*";

Potestative, which attribute powers. For example, "A worker has the power to terminate his work contract";

Evidentiary, which establish the conclusion to be drawn from certain evidence. For example, "It is presumed that dismissal was discriminatory";

Existential, which indicate the beginning or the termination of the existence of a legal entity. For example, "The company ceases to exist";

Norm-concerning effects, which state the modifications of norms such as abrogation, repeal, substitution, and so on.

[6] Note that these constructions can give rise to very complex rule dependencies, because we can have that 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.

Persistence of normative effects [20]. Some normative effects persist over time unless some other and subsequent event terminate them. For example: "If one causes damage, one has to provide compensation.". Other effects hold on the condition and only while the antecedent conditions of the rules hold. For example: "If one is in a public office, one is forbidden to smoke".

Values [4]. Usually, some values are promoted by the legal rules. Modelling rules sometimes needs to support the representation of *values* and *value preferences*, which can play also the role of meta-criteria for solving rule conflicts. (Given two conflicting rules r_1 and r_2, value v_1, promoted by r_1, is preferred to value v_2, promoted by r_2, and so r_1 overrides r_2.)

An interesting question is whether rule interchange languages for the legal domain should be expressive enough to fully model all the features listed above, or whether some of these requirements can be meet at the reasoning level, at the level responsible for structuring, evaluating and comparing legal arguments constructed from rules and other sources. The following sections will consider this issue when discussing these requirements in the context of some existing rule interchange formats: RuleML, SBVR, SWRL, RIF, and LKIF.

3 Overview of Some Rule Interchange Languages

3.1 The Rule Markup Language (RuleML)

RuleML[7] is an XML based language for the representation of rules. It offers facilities to specify different types of rules from derivation rules to transformation rules to reaction rules. It is capable of specifying queries and inferences in Web ontologies, mappings between Web ontologies, and dynamic Web behaviours of workflows, services, and agents [8]. RuleML was intended as the canonical web language for rules, based on XML markup, formal semantics and efficient implementations. Its purpose is to allow exchange of rules between major commercial and non-commercial rules systems on the Web and various client-server systems located within large corporations to facilitate business-to-customer (B2C) and business-to-business (B2B) interactions over the Web.

RuleML provides a way of expressing business rules in modular stand-alone units. It allows the deployment, execution, and exchange of rules between different systems and tools. It is expected that RuleML will be the declarative method to describe rules on the Web and distributed systems [47]. RuleML arranges rule types in an hierarchical structure comprising reaction rules (event-condition-action-effect rules), transformation rules (functional-equational rules), derivation rules (implicational-inference rules), facts ('premiseless' derivation rules, i.e., derivation rules with empty bodies), queries ('conclusionless' derivation rules, i.e., derivation rules with empty heads) and integrity constraints (consistency-maintenance rules). Each part of a rule is an expression that has specific functions in the rule. The RuleML Hierarchy first directly branches out into two categories: Reaction Rules and Transformation Rules. Transformation Rules then break

[7] `http://www.ruleml.org`

down into Derivation Rules, that, in turn, subdivide into Facts and Queries. Finally, Queries break down into Integrity Constraints [36].

The way RuleML achieves flexibility and extensibility is based on the use and composition of modules. Each module is meant to implement a particular feature relevant for a specific language or application (e.g., modules for various types of negation, for example, classical negation, and negation as failures). Each module is intended to refer to a semantic interpretation of the feature implemented in the module. However, RuleML does not have a mechanism to specify semantic structures on which to evaluate elements of the language.

The key strength of Rules is its extensibility. Thus despite that currently there is no dialect specifically intended for the representation of legal rules a few works proposed extension and interpretation for this area, in particular for the representation of (business) contracts [21, 16, 18].

The contribution of [21] by Grosof was the proposal of adopting courteous logic programming (a variant of defeasible logic) as execution model for RuleML rule-base corresponding to the clauses of a contract. Accordingly, Grosof's proposal meets the defeasiblity key requirement for modelling legal rules. Technically [21] uses derivation rules, but then a courteous logic program implemented as Sweet Jess rules constitutes executable specifications, where the conclusion of a rule can be executed by a computer program producing effects. Thus the approach bridges the gap among the various types of rules in the RuleML family.

The limitation of [21] is that it does not consider normative effects (i.e., it is not possible to differentiate between obligations and permissions). This limitation has been addresses by Governatori [16], where defeasible logic is extended with the standard deontic operators for obligations, permissions and prohibitions as well as a new special deontic operator to model violations and penalties for the violations. Furthermore [16] distinguishes between constitutive and prescriptive rules. It provides a RuleML compliant DTD for representing the various deontic elements, and discusses various options for the modelling of such notions in defeasible logic. [18] implements [16], in a Semantic Web framework with support for RDF databases, to provide an environment to model, monitor and perform business contracts.

The modelling approach proposed in [16] has proven successful for various legal concepts (for example the legal notion of trust [34]) and it has been extended to cover temporal aspects [20] norm dynamics [19], and it has been applied to the study of business process compliance [17].

3.2 Semantics of Business Vocabulary and Business Rules (SBVR)

SBVR [39] is a standard proposed by the Object Managament Group (OMG) for the representation and formalisation of business ontologies, including business vocabularies, business facts and business rules. The main purpose of SBVR is to give the basis for formal and detailed natural language declarative specifications of business entities and policies. It provides a way to represent statements in controlled natural language as logic structures called semantic formulations. The formal representation is based on several logics including first order logic, alethic modal logic and deontic logic, furthermore it adopts model theoretic interpretations for semantic formulations. It is worth noticing that the focus of SBVR is on modelling not providing a framework for executing the rules.

The two most relevant and salient features of SBVR for the modelling of norms are the introduction of deontic operators to represent obligations and permissions and the use of controlled natural languages for modelling norms. These two features, combined with the underlying formalisation, make SBVR a conceptual language able to capture some of the requirements discussed in Section 2. In particular the requirements about the structural isomorphism and the ability to capture some normative effects.

Unfortunately, the semantics for the deontic modalities is left underspecified and the proposed interpretation suffers from some drawbacks to model norms. First of all it assumes formulas like Barcan formula and its converse that allow for the permutation of universal quantifiers and alethic modalities (i.e., $\Box\forall x\phi(x) \equiv \forall x\Box\phi(x)$). The main consequence for this is that it then forces the used of possible world models with constant domains. While this assumption seems to be harmless, it has some important consequences for the modelling of norms. Recent literature on deontic logic (see, among others, [40]) agrees that normal deontic logics –that is logics that admit necessitation (i.e., from $\vdash \phi$ derive $\vdash O\phi$, where O is the deontic modality for obligation)– are not suitable to model norms. However, any deontic logic based on possible world semantics with constant domains, and having at least one genuine obligation is a normal deontic logic [46]. Thus an adequate model theoretic semantics for the deontic modalities seems problematic. Another problem caused by standard deontic logic is that of contrary-to-duty obligations, i.e., obligations arising from violations of other obligations. It is well known that these cannot be handled properly by standard deontic logics [9]. However, as [16] points out these are frequent in legal documents, and contracts in particular. [39] recognises this limitation and the issue of handling this is left for future versions of the specifications.

SBVR suggests the equivalence $(\phi \rightarrow O\psi) \equiv O(\phi \rightarrow \psi)$ to transform semantic expressions having the deontic operator not as main operator into an expression where the deontic operator is the main operator. The proposed transformation imposes additional non-standard constraints on possible world semantics; moreover the proposed transformation poses some concerns on its conceptual soundness since typically the deontic modality applies just to the conclusion of the rule or to the conditional corresponding to the rule (see the discussion about prescriptive rules in Section 2).

The final drawback of the proposed semantics for SBVR is that, being based on classical first order logic it is not suitable to handle conflicts. But as we have highlighted in the discussion of the requirements, handling conflicts is one of the key features if one wants to use rule to reason with legal rules.

3.3 The Semantic Web Rule Language (SWRL)

The Semantics Web Rule Language (SWRL) is a W3C proposal for a rule interchange format which combines ontologies represented in the Description Logic (DL) subset of OWL with an XML format for rules in the Unary/Binary Datalog subset of the Rule Markup Language (RuleML).[8] While both OWL-DL and Datalog, separately, are de-

[8] `http://www.w3.org/Submission/SWRL/`

cidable subsets of first-order logic, the union OWL-DL and Datalog, as in SWRL, is undecidable.

Three approaches to implementing inference engines for SWRL have been tried. In Hoolet, SWRL files are translated into a language for full first-order logic and a general purpose first-order theorem prover is used to derive inferences, with all the undesirable computational properties this entails. In Bossam, SWRL files are translated into rules for a forward-chaining production rule system[9]. This procedure translates OWL-DL axioms into rules, but with a loss of information, since some information expressable in OWL-DL axioms cannot be represented in Bossam's production rule language. Thus the resulting inference engine with this approach is incomplete. Finally, a third approach, taken by Pellet, is to start with tableaux theorem-prover for OWL-DL and extend this to support the "DL-safe" subset of SWRL [42].

Let us now try to evaluate SWRL with respect to the requirements we have identified for modeling and reasoning with legal rules. Since SWRL rules are Horn clauses, it is not possible to model legal rules in an isomorphic way. Most legal rules would need to be modeled using several SWRL rules. Morever, the lack of negation in Horn clause logic is a problem, since both the condtions and conclusions of legal rules are often negated. Perhaps this can be overcome in SWRL to some extent by defining complementary predicates using OWL classes. Since rules are represented in XML in SWRL, they can be reified by giving them identifiers using XML attributes. Similarly, the various properties of legal rules, such as their validity, could presumably also be represented using XML attributes. But since these attributes would be at a meta-level, outside the formal syntax and semantics of the SWRL logic, and since SWRL inherits the monotonic semantics of classical first-order logic, it is not clear how these measures could be used to resolve conflicts among legal rules, using principals like *lex superior* or to reason with exclusionary rules. A further problem is that SWRL provides no standard way to annotate the conditions of rules with information about the distribution of the burden proof, but it should be possible to extend SWRL, again using XML attributes, to provide this information. Semantically, unlike legal rules SWRL rules do contrapose, since they are interpreted as material conditionals of classical logic, but in practice SWRL reasoners are too weak to derive the undesired conclusions. Morever, even if the reasoners were stronger, without some way to represent negative facts, *modus tollens* would never be applicable. If we separate the syntax of SWRL from its semantics, it might be possible to develop a nonmontonic logic which solves some of these problems, while retaining SWRL's syntax, but with some additional XML attributes for annotating rules. But it is difficult to imagine how this approach could satisfy the isomorphism requirement.

3.4 The Rule Interchange Format (RIF)

The Rule Interchange Format (RIF) Working Group of the World-Wide-Web Consortium was established in 2005, about a year afer the SWRL proposal was submitted, with the goal of developing an extensible rule interchange format for the Web, building on

[9] `http://owl.man.ac.uk/hoolet/`

prior experience in related initiatives and W3C submissions, included RuleML, SWRL, Common Logic and SBVR, among others.[10]

Like RuleML, RIF is intended to be an extensible framework for a whole family of rule languages, possibly with different semantics. Currently, RIF consists of draft reports for several components, including the following:

RIF Core. Defines an XML syntax for definite Horn rules without function symbols, i.e Datalog, with a standard first-order semantics.

RIF Basic Logic Dialect (RIF-BLD). Defines a language, building on RIF Core, for definite Horn rules with equality and a standard first-order semantics.

RIF Production Rule Dialect (RIF-PRD). Extends RIF Core to define a language for production rules, i.e condition-action rules in which the actions supported are limited to modifications of the facts in working memory.

RIF RDF and OWL Compatibility. Defines semantics for the integrated use of RIF, RDF and OWL in applications.

RIF Framework for Logic Dialects (RIF-FLD). Defines a framework which may be used to configure RIF dialects. For example, one can choose whether negation is interpreted classically or negation-as-failure, as in logic programming, or whether rules are interpreted as material implications or inference rules.

For the purpose of representing legal rule, RIF Core and RIF-BLD both appear to suffer from the same problems as SWRL, and for the same reasons. The production rule dialect of RIF does not seem relevant, since production rules, with their ability to delete information from working memory, are a procedural programming paradigm which may or may not be useful for implementing a legal reasoning support system, but which are not suitable as a language for modeling legal norms. An interesting question is whether the RIF Framework for Logic Dialects (RIF-FLD) could be used to configure a RIF dialect which is more suitable for modeling legal norms. Although this question requires further research, our initial impression is that the space of configurations possible is limited subsets of first-order logic and well-known logic programming paradigms. Languages suitable for modeling legal norms presumably fall outside of this space, since neither first-order logic nor common logic programming languages provide sufficient support isomorphic modeling of legal, allocating the burden of proof among the parties in a legal dispute, or modeling principals for resolving conflicts, such as *lex superior* for resolving rule conflicts.

3.5 The Legal Knowledge Interchange Format (LKIF)

The Legal Knowledge Interchange Format (LKIF) was developed in a three-year European research project, ESTRELLA[11], which completed its work at the end of 2008 [12, 11]. The goal of the ESTRELLA project, with respect to LKIF, was to develop an interchange format for formal models of legal norms which is sufficient for modeling legal knowledge in a broad range of application scenarios, builds on existing standards,

[10] `http://www.w3.org/2005/rules/wg/charter.html`
[11] IST-4-027655

especially in the context of the Semantic Web, and informed by the state-of-the-art of the field of Artificial Intelligence and Law.

LKIF is an XML Schema for representing theories and arguments constructed from theories. A theory in LKIF consists of a set of axioms and defeasible inference rules. The language of individuals, predicate and function symbols used by the theory can be imported from an ontology represented in the Web Ontology Language (OWL). Importing an ontology also imports the axioms of the ontology. All symbols are represented using Universal Resource Identifiers (URIs). Other LKIF files may also be imported, enabling complex theories to modularized.

Axioms are named formulas of full first-order logic. The heads and bodies of inference rules are sequences of first-order formulas. All the usual logical operators are supported and may be arbitrarily embedded: disjunction ($\wedge$), conjunction ($\vee$), negation ($\neg$), material implication ($\rightarrow$) and the biconditional ($\leftrightarrow$). Both existential ($\exists$) and universal ($\forall$) quantifiers are supported. Free variables in inference rules represent schema variables.

Terms in formulas may be atomic values or compound expressions. Values are represented using XML Scheme Definition (XSD) datatypes. Atomic formulas are reified and can be used as terms, allowing some meta-level propositions to be expressed.

The schema for atomic formulas has been designed to allow theories to be displayed and printed in plain, natural language, using Cascaded Style Sheets (CSS). An atomic proposition may first be represented in propositional logic, using natural language, and later enriched to become a first-order model, by marking up the variables and constants of the proposition and specifying its predicate using an XML attribute. This feature of LKIF is essential for enabling domain experts, not just computer specialists, to write and validate theories.

Support for allocating the burden of proof when constructing arguments from theories in dialogues is provided. An *assumable* attribute is provided for atomic formulas, to indicate they may be assumed true until they have been challenged or questioned. An *exception* attribute is provided for negated formulas, to indicate that P may be presumed not true *unless* P has been proven. This is similar to negation-as-failure in logic programming, in that the failure to find a proof for P is sufficient to prove $\neg P$, if $\neg P$ is an exception.

Arguments in LKIF link a sequence of premises to a conclusion, where both the premises and the conclusion are atomic formulas. Attributes are provided for stating the direction of the argument (pro or con), the argumentation scheme applied and the role of each premise in an argument. Arguments can be linked together to form argument graphs. The legal proof standard each proposition at issue must satisfy, such as "preponderance of the evidence" or "beyond reasonable doubt" may be specified. Attributes are provided for recording the relative weight assigned to each argument by the finder of fact, such as the jury, or some other audience, as well as the status of each issue in the proceeding.

All of the main elements of an LKIF file may be assigned Universal Resource Identifiers, allowing them to be referenced in other documents, anywhere on the World Wide Web. Cross references between elements of legal source documents and the elements of the LKIF document which model these sources may be included within the LKIF file, using a sequence of *source* elements. The scheme allows *m* to *n* relationships between legal sources and elements of the LKIF model to be represented.

LKIF builds on and uses many existing World Wide Web standards, including the XML, Universal Resource Identifiers, XML Namespaces, the Resource Description Framework (RDF) and the Web Ontology Language (OWL). However, for a variety of reasons it does not use other XML schemas for modeling legal rules, such as Common Logic, RuleML, the Semantic Web Rule Language (SWRL), or the Rule Interchange Format (RIF). Common Logic is an ISO standard for representing formulas of first-order classical logic. While LKIF includes a sublanguage for first-order logic, LKIF has been designed to allow formulas of first-order logic to be represented in human readable form in natural language, to ease development, maintenance and validation by domain experts. Moreover, the ISO Common Logic standard does not look like it will be widely adopted within the World Wide Web community, which has its own standards body, the World Wide Web Consortium. RuleML, SWRL and RIF, among other efforts, are competing to become the Web standard for rules. At the beginning of the ESTRELLA project, SWRL was the leading candidate. In the meantime, during the development of LKIF in Estrella, RIF has become the leading contender. But neither SWRL nor RIF are currently expressive enough for the legal domain. Legal rules can be understood as domain-dependent defeasible inference rules. They cannot be adequately modeled as material implications in first-order logic. However, an LKIF theory can in principal import a first-order theory represented in any XML format, to be used as part of the axioms of the theory. This feature of LKIF enables a part of the legal theory to be represented in first-order logic, using whatever format eventually becomes the World Wide Web standard.

A reference inference engine for LKIF, called Carneades, was developed in ESTRELLA [13]. Carneades is written in a functional style, using the Scheme programming language, and is available as Open Source software.[12]. Carneades places some restrictions on LKIF rules: The heads of rules are limited to literals (postive or negated atomic formulas) and the biconditional ($\leftrightarrow$) operator and first-order quantifiers are not supported. (Free variables, represented schema variables, are supported.) Since LKIF is a very expressive language, the computational complexity of various reasoning tasks can be high, depending on which features of the language have been used in a model. Carneades allows programmers to choose a search strategy (depth-first, breadth-first, iterative deepening), and to develop and plug-in custom, heuristic search strategies. Resource bounds can be set to assure that every search for arguments about an issue terminates in a predictable period of time.

Because of the open-ended nature of legal reasoning, no formal model of a legal domain, in any logic, can guarantee that inferences are legally correct in some absolute sense. The formal model may be incorrect or incomplete. Or the search space may be so large as to make the legal problem undecidable or intractable. Thus legal reasoning and argumentation necessarily has a procedural component. Legal procedures are designed to assure that justifiable decisions can be made in finite time, expending limited resources, as in Loui's conception of resource-bounded, non-demonstrative reasoning [25]. LKIF and Carneades are designed for use in such procedures. The ability

[12] `http://carneades.berlios.de`

of Carneades to generate arguments, making the reasoning of the system transparent and auditable, are essential for documenting and justifying legal decisions.

4 Conclusions

In this paper we outlined a comprehensive list of requirements for rule interchange languages for applications in the legal domain. We used these requirements to assess the suitability of some rule interchange languages, such as RuleML, SBVR, SWRL and RIF, for modeling legal rules. We finally presented the Legal Knowledge Interchange Format (LKIF), a new rule interchange format developed specifically for legal applications.

Currently, there is no language, among those that we have examined here, which satisfy all the requirements we have listed in Section 2: all languages have thus their pros and cons. It should be noted, however, that not all those requirements play the same role in the legal domain. While the concept of defeasibility, for example, is almost ubiquitous in the law, others, such as the representation of some temporal properties (in particular, the time when a rule is in force) are definitely more important when we are dealing, e.g., with legislation.

Accordingly, it seems to us that some languages are not currently expressive enough for the legal domain. In particular, RIF and SWRL fail to meet the defeasibility requirement, which is quite fundamental: legal rules are often defeasible and cannot be correctly represented through material implications in first-order logic. Hence, LKIF and RuleML look suitable and more flexible in this regard. Another requirement, among others, which seems crucial for modeling legal rules is the correct representation of the many different types of normative effects, and the need to capture, for example, the deontic concepts. Here, SBVR and RuleML [16], though with some limitations, show how to do that in a rather satisfactory way.

Finally, it should remarked that, for specific types of application, some (but not all) of these requirements can be somehow relaxed. For example, strict isomorphism is not always compulsory if we have to devise a system for monitoring norm compliance but we do not develop additional in-house rules for normalising legal norm, namely, for identifying formal loopholes, deadlocks and inconsistencies, and making hidden conditions (such as chains of reparative obligations) explicit. Without such a mechanism, it may hard to guarantee that a given process is compliant, because we do not know if all relevant norms have been considered. Anyway, we do believe that most of the requirements seem fundamental for representing legal rules.

Acknowledgements

We would like to thank Harold Boley and Monica Palmirani for their valuable comments on earlier versions of this paper.

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

1. F. Abate and E. J. Jewell, editors. *New Oxford American Dictionary*. Oxford University Press, 2001.
2. Architecture for Knowledge-Oriented Management of African Normative Texts using Open Standards and Ontologies. `http://www.akomantoso.org/`, 2009.
3. T. Arnold-Moore. Automatic generation of amendment legislation. In *Proc. ICAIL'97*, New York, 1997. ACM.
4. T. Bench-Capon. The missing link revisted: The role of teleology in representing legal argument. *Artificial Intelligence and Law*, 10(1-3):79–94, September 2002.
5. T. Bench-Capon and F. Coenen. Isomorphism and legal knowledge based systems. *Artificial Intelligence and Law*, 1(1):65–86, 1992.
6. 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-Verlag, 2005.
7. A. Boer, R. Hoekstra, and R. Winkels. Metalex: Legislation in XML. In *Proc. JURIX 2002*, Amsterdam, 2002. IOS Press.
8. H. Boley, S. Tabet, and G. Wagner. Design rationale for RuleML: A markup language for Semantic Web rules. In I. F. Cruz, S. Decker, J. Euzenat, and D. L. McGuinness, editors, *Proc. SWWS'01, The first Semantic Web Working Symposium*, pages 381–401, 2001.
9. J. Carmo and A. J. Jones. Deontic logic and contrary to duties. In D. Gabbay and F. Guenther, editors, *Handbook of Philosophical Logic. 2nd Edition*, volume 8, pages 265–343. Kluwer, Dordrecht, 2002.
10. ChLexML. `http://www.svri.ch/`, 2009.
11. ESTRELLA Project. Estrella user report. Deliverable 4.5, European Commission, 2008.
12. ESTRELLA Project. The legal knowledge interchange format (LKIF). Deliverable 4.3, European Commission, 2008.
13. ESTRELLA Project. The reference LKIF inference engine. Deliverable 4.3, European Commission, 2008.
14. S. Goedertier and J. Vanthienen. A declarative approach for flexible business. In J. Eder and S. Dustdar, editors, *BPM Workshops 2006*, pages 5–14, Berlin, 2006. Springer.
15. T. F. Gordon. *The Pleadings Game; An Artificial Intelligence Model of Procedural Justice*. Springer, New York, 1995. Book version of 1993 Ph.D. Thesis; University of Darmstadt.
16. G. Governatori. Representing business contracts in RuleML. *International Journal of Cooperative Information Systems*, 14(2-3):181–216, 2005.
17. G. Governatori, Z. Milosevic, and S. Sadiq. Compliance checking between business processes and business contracts. In *Proc. EDOC 2006*, pages 221–232. IEEE, 2006.
18. G. Governatori and D. H. Pham. Dr-contract: An architecture for e-contracts in defeasible logic. *International Journal of Business Process Integration and Management*, 5(4), 2009.
19. G. Governatori and A. Rotolo. Changing legal systems: Legal abrogations and annulments in defeasible logic. *The Logic Journal of IGPL*, forthcoming.
20. G. Governatori, A. Rotolo, and G. Sartor. Temporalised normative positions in defeasible logic. In *Proc. ICAIL'05*, pages 25–34. ACM Press, 2005.
21. B. Grosof. Representing e-commerce rules via situated courteous logic programs in RuleML. *Electronic Commerce Research and Applications*, 3(1):2–20, 2004.
22. J. C. Hage. *Reasoning with Rules – An Essay on Legal Reasoning and its Underlying Logic*. Kluwer Academic Publishers, Dordrecht, 1997.
23. H. Kelsen. *General theory of norms*. Clarendon, Oxford, 1991.
24. Legal and Advice Sectors Metadata Scheme (LAMS5).
`http://www.lcd.gov.uk/consult/meta/metafr.htm`.

25. R. P. Loui. Process and policy: resource-bounded non-demonstrative reasoning. *Computational Intelligence*, 14:1–38, 1998.
26. C. Lupo and C. Batini. A federative approach to laws access by citizens: The Normeinrete system. In R. Traunmuller, editor, *Proc. Second International Conference on Electronic Government*, Berlin, 2003. Springer.
27. 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. Technical report, IST-2004-027655 ESTRELLA European project for Standardised Transparent Representations in order to Extend Legal Accessibility: Deliverable 3.1, 2007.
28. J. McClure. Legal-rdf vocabularies, requirements and design rationale. In *Proc. V Legislative XML Workshop*, Florence, 2006. European Press.
29. The OWL services coalition: OWL-S 1.2 pre-release. `http://www.ai.sri.com/daml/services/owl-s/1.2/`, 2006.
30. A. Paschke, M. Bichler, and J. Dietrich. Contractlog: An approach to rule based monitoring and execution of service level agreements. In A. Adi, S. Stoutenburg, and S. Tabet, editors, *RuleML 2005*, pages 209–217, Berlin, 2005. Springer.
31. M. Pesic and W. van der Aalst. A declarative approach for flexible business. In J. Eder and S. Dustdar, editors, *BPM Workshops 2006*, pages 169–180, Berlin, 2006. Springer.
32. 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.
33. D. Roman, U. Keller, H. Lausen, J. de Bruijn, R. Lara, M. Stollberg, A. Polleres, C. Feier, C. Bussler, and D. Fensel. Web service modeling ontology. *Applied Ontology*, 1(1):77–106, 2005.
34. A. Rotolo, G. Sartor, and C. Smith. Good faith in contract negotiation and performance. *International Journal of Business Process Integration and Management*, 5(4), 2009.
35. R. Rubino, A. Rotolo, and G. Sartor. An OWL ontology of fundamental legal concepts. In *Proc. JURIX 2006*, pages 101–110, 2006.
36. RuleML. The Rule Markup Initiative. `http://www.ruleml.org`, 20th August 2009.
37. S. Sadiq, M. Orlowska, and W. Sadiq. Specification and validation of process constraints for flexible workflows. *Information Systems*, 30(5):349–378, 2005.
38. 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. Springer, 2005.
39. OMG: Semantics of business vocabulary and business rules (SBVR). `http://www.businessrulesgroup.org/sbvr.shtml`, 2008.
40. M. Sergot. A computational theory of normative positions. *ACM Transactions on Computational Logic*, 2(4):581–622, 2001.
41. M. Sergot, F. Sadri, R. Kowalski, F. Kriwaczek, P. Hammond, and H. Cory. The British Nationality Act as a logic program. *Communications of the ACM*, 29(5):370–386, 1986.
42. E. Sirin, B. Parsia, B. Grau, A. Kalyanpur, and Y. Katz. Pellet: A practical OWL-DL reasoner. *Web Semantics*, 5(2):51–53, 2007.
43. W. van der Aalst, M. Weske, and D. Grünbauer. Case handling: a new paradigm for business process support. *Data Knowledge Engineering*, 53(2):129–162, 2005.
44. B. Verheij. *Rules, Reasons, Arguments. Formal Studies of Argumentation and Defeat*. Ph.d., Universiteit Maastricht, 1996.
45. G. H. von Wright. *Norm and Action*. Routledge, London, 1963.
46. G. Waagbø. Quantified modal logic with neighborhood semantics. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 38:491–499, 1992.
47. G. Wagner, G. Antoniou, S. Tabet, and H. Boley. The abstract syntax of RuleML – towards a general web rule language framework. In *Proc. Web Intelligence 2004*, pages 628–631. IEEE, 2004.