Representing E-Business Rules for the Semantic Web: Situated Courteous Logic Programs in RuleML*

Benjamin N. Grosof
MIT Sloan School of Management
bgrosof@mit.edu ; http://www.mit.edu/~bgrosof/

Abstract

We give an overview of current efforts to standardize e-business rules knowledge representation (KR) in XML as part of the Semantic Web. The focus especially on the design approach and criteria of RuleML, an emerging standard that we are helping to lead. We discuss the issues of standardization and Webizing which RuleML addresses. Finally, we describe our current effort to prototype tools for communication and inferencing of e-business rules represented in RuleML. Specifically, we give examples of situated courteous logic programs, an expressively general portion of RuleML that supports prioritized conflict handling and procedural attachments for actions and queries. **NB**: for a longer version of this paper, and related material, see the author's website.

1 Introduction

In this paper, we give an overview of our current efforts to standardize rules knowledge representation (KR) in XML. We focus especially on the design approach and criteria of RuleML, an emerging standard that we are helping to lead. We view this as part of the R&D community's effort to develop the *Semantic Web*, a vision of moving the Web to support program-to-program communication of data that has high-level semantics that are shared by both parties (programs) in communication. RuleML is based on a fundamental rule KR, declarative *logic programs* (LP). In previous work we have expressively extended LP with features for prioritized conflict handling and procedural attachments to perform actions and queries; the result is called *situated courteous* logic programs (SCLP). In previous work we have also developed BRML, an XML syntax for SCLP. We discuss BRML's limitations – it only shallowly *Webized* SCLP.

The novel contribution of this paper is to overview RuleML, the issues of standardization and *Webizing* which it addresses, and our current efforts at developing tools for it. In particular, we are prototyping tools for communication and inferencing of e-business rules represented in RuleML. Specifically, we give an example of situated courteous logic programs, an expressively general portion of RuleML.

*In Proc. 2001 Wksh. on Information Technologies and Systems (WITS '01) (http://www.busi.mun.ca/parsons/wits2001/)

2 RuleML and expressively extended Logic Programs

We are leading, with Harold Boley of DFKI (Germany) and Said Tabet of Nisus Inc. (USA), an early-phase standards effort on a markup language for exchange of rules in XML, called RuleML (Rule Markup Language)[1]. The goal of this effort is eventual adoption as a Web standard, e.g., via the World Wide Web Consortium (W3C) within its new Semantic Web Activity.[2] Along the way there are a number of interesting new research issues.

RuleML is, at its heart, an XML syntax for rule knowledge representation (KR), that is inter-operable among major commercial rule systems. It is especially oriented towards four currently commercially important ("CCI") families of rule systems: SQL (relational database), Prolog, production rules (cf. OPS5, CLIPS, Jess) and Event-Condition-Action rules (ECA). These kinds of rules today are often found embedded in systems built using Object-Oriented (OO) programming languages (e.g., C++ and Java), and are often used for business process connectors / workflow. These four families of rule systems all have common core abstraction: declarative logic programs (LP) ([1] provides a helpful review). "Declarative" here means in the sense of KR theory.[3] Note that this supports both backward inferencing and forward inferencing.

In addition to specifying XML syntax for rules, RuleML specifies an associated KR semantics (KRsem). The KRsem specifies what set of conclusions are sanctioned for any given set of premises. Being able to define an XML syntax is relatively straightforward. Crucial is the semantics (KRsem) and the choice of expressive features.

The kernel representation in RuleML is: Horn declarative logic programs. Extensions to this representation are defined for several additional expressive features:

- negation: negation-as-failure and classical negation;
- prioritized conflict handling: e.g., cf. *courteous* logic programs [3];
- disciplined procedural attachments for queries and actions: e.g., cf. *situated* logic programs [2];

and other features as well. In addition, RuleML defines some useful expressive restrictions (e.g., Datalog, facts-only, binary-relations-only), not only expressive generalizations.

In January 2001, we released a first public version of a family of DTD's for several flavors of rules in RuleML. This was presented at the W3C's Technical Plenary Meeting[4] held Feb. 26 to Mar. 2, 2001. Especially since then, RuleML has attracted a considerable degree of interest in the R&D community. Meanwhile, the design has been evolving to further versions.

RuleML largely grows out of the design approach and design criteria of Business Rules Markup Language (BRML) which was developed in our previous work at IBM Research and which is implemented in IBM CommonRules[5] available under free trial license from

[1] http://www.dfki.de/ruleml and http://www.mit.edu/~bgrosof/#RuleML

[2] http://www.w3.org/2001/sw

[3] in which: a given set of premises entails a set of sanctioned conclusions, independent of inferencing control strategy or procedural aspects, e.g., independent of whether inferencing direction is goal-directed query answering ("backward") vs. data-driven ("forward").

[4] a large convocation of most of its face-to-face standards working group meetings

[5] http://www.research.ibm.com/rules and http://alphaworks.ibm.com

IBM alphaWorks. The design approach and design criteria of CommonRules and BRML are described in [3], and in the documentation in the CommonRules download package. BRML's expressive class is situated courteous logic programs, i.e., declarative logic programs with negation-as-failure, (limited) classical negation, prioritized conflict handling, and disciplined procedural attachments for queries and actions.

SCLP and BRML were developed in large part to surmount the limitations of the first major attempt at a KR interlingua: Knowledge Interchange Format (KIF)[6]. KIF was developed in the early 1990's as a system for researchers, as opposed to commercial applications, to exchange logical-form knowledge. KIF's KR is essentially classical logic. It has not yet become widely used for deployed commercial applications. In particular, it has two major limitations that prevent it from representing e-business rules of the kind used in CCI rule applications. Firstly, it is *pure-belief*; it cannot represent/specify procedural attachments for queries or actions (or for anything else!). Secondly, it is *logically monotonic*; it cannot represent/specify negation-as-failure or prioritized conflict handling which are logically *non*-monotonic.[7] Yet procedural attachments and logical non-monotonicity are heavily used in CCI rule systems and their applications. Example kinds of non-monotonicity include: priority between rules in Prolog based on static rule sequence; dynamically-computed priorities among rules in production rule and ECA rule systems; inheritance with exceptions; and updating in databases (where more recent assertions override previous ones).

Another advantage of (situated courteous) logic programs is computational scaleability: inferencing is tractable (worst-case polynomial-time) for a broad expressive case: e.g., when the number of logical variables per rule is bounded, and logical functions are disallowed; classical logic. By contrast, classical logic inferencing is NP-hard for this case.

RuleML differs in several significant respects from its BRML predecessor, however. These differences largely revolve around "Webizing" the KR:

- URI's[8] for logical vocabulary and knowledge subsets
- labels for rules/rulebases, import/export
- headers: meta-data describes the XML document's expressive class
- procedural attachments using Web protocols/services; queries or actions via CGI/servlets/SOAP/...

Such Webizing, and interoperability of KR on the Web, involve several kinds of practical mechanics beyond the representation proper. These include to:

- build on existing W3C standards: namespaces, ...
- share mechanisms with other emerging/extant standards for KR and ontologies on the Web, including especially RDF/RDFS[9] and DAML+OIL[10], the ontology representation that is the main technical point of departure for the newly-formed WebOnt Working Group of the W3C

[6]http://logic.stanford.edu/kif and http://www.cs.umbc.edu/kif/

[7]A particular KR is said to be logically monotonic when it has the property that adding premises never results in retracting (sanctioned) conclusions.

[8]Uniform Resource Identifiers, a generalization of Uniform Resource Locators (URL's), a W3C standard

[9]http://www.w3.org, search for Resource Description Format (RDF) and RDF Schema

[10]http://www.daml.org

- use ontologies for rules, and rules for ontologies
- support ontology tags in: rulebase, predicate symbol, ...

RuleML has some first steps of Webizing rule KR, including:

- URI's for logical vocabulary and knowledge subsets: predicates, functions, rules, rulebases, labels for rules and rulebases
- facilitation of an (alternative) RDF syntax: by having its XML syntax mostly avoid reliance on ordered-ness of child elements within any element (there are already some partial first drafts of such an alternative RDF syntax)
- support of object-oriented style argument "roles" (cf. named members of a Java/C++ class; rather than simply relying on position within an ordered tuple of arguments in the manner of, say, Prolog)

3 Prototype of Situated Courteous Logic Programs in RuleML

We are currently prototyping for the first time how to specify courteous LP and situated LP as part of RuleML, and how to perform inferencing and translation for situated courteous LP in RuleML.[11] Our tools are implemented using XSLT[12] (a tool for transforming from one XML format/syntax into another) and Java, and make use of IBM CommonRules as a middle component. The prototype tools implement both forward and backward inferencing for SCLP RuleML. For backward inferencing, we make use of the XSB[13] logic programming system. The inferencing tools take SCLP RuleML premises as input and return SCLP RuleML conclusions as output. The tools also implement translation of RuleML to/from several other rule system representations, including Prolog, Knowledge Interchange Format (KIF), and another XML format. We are using the tools to run several examples taken from e-business application domains, including supply chain leadtime, e-bookstore pricing, and creditworthiness. Next, we give an example of supply chain leadtime.

In business-to-business commerce, e.g., in manufacturing supply chains, sellers often specify how much lead time, i.e., minimum advance notice before scheduled delivery date, is required in order to place or modify a purchase order. An example of a parts supplier vendor's lead time policy is:

• (Rule A:) "14 days lead time if the buyer is a preferred customer."
• (Rule B:) "30 days lead time if the ordered item is a minor part."
• (Rule C:) "2 days lead time if: the ordered item's item-type is backlogged at the vendor, and the order is a modification to reduce the quantity of the item, and the buyer is a preferred customer."
• (Priority Rule:) "If Rules A and C both apply, then Rule C 'wins', i.e., 2 days lead time."

The rationale for Rule C is that the vendor is having trouble filling its overall orders (from all its buyers) for the item, and thus is pleased to have this buyer relieve the pressure.

[11] See our related short paper describing the prototype, in the WITS-01 proceedings.
[12] see, e.g., http://www.w3.org/TR/xslt
[13] http://www.sunysb.edu/~sbprolog

Rules A, B, and C may conflict: two or three of them might apply to a given purchase order. The Priority Rule provides partial prioritization information – its rationale might be that Rule C is more specific, or more recent, than Rule A. However, the above rule-set leaves unspecified how to resolve conflict between Rules A and B, for example; no relative priority between them is specified as yet. This reflects a common situation when rules accumulate over time, or are specified by multiple authors: at any given moment during the process of incremental specification, there may be insufficient justified priority to resolve all potential conflicts.

The above example can be straightforwardly represented in CLP as follows:

$\langle a \rangle \quad orderModificationNotice(?Order, days14)$
$\quad \leftarrow \; preferredCustomerOf(?Buyer, ?Seller) \wedge$
$\quad\quad purchaseOrder(?Order, ?Buyer, ?Seller).$

$\langle b \rangle \quad orderModificationNotice(?Order, days30)$
$\quad \leftarrow \; minorPart(?Order) \wedge$
$\quad\quad purchaseOrder(?Order, ?Buyer, ?Seller).$

$\langle c \rangle \quad orderModificationNotice(?Order, days2)$
$\quad \leftarrow \; preferredCustomerOf(?Buyer, ?Seller) \wedge$
$\quad\quad orderModificationType(?Order, reduce) \wedge$
$\quad\quad orderItemIsInBacklog(?Order) \wedge$
$\quad\quad purchaseOrder(?Order, ?Buyer, ?Seller).$

The rule labels, e.g., a, at the left of each rule above are used as handles/names for specifying *prioritization* (partial-) ordering via the syntactically reserved predicate $overrides$ which indicates that its first argument is higher priority than its second argument. $overrides$ is otherwise an ordinary predicate, e.g., $overrides$ facts can be inferred via rules.

$overrides(c, a).$

The scope of what constitutes conflict is specified by *mutual exclusion (mutex) statements*, which can be viewed as a kind of integrity constraint. Each such statement says that is a contradiction/inconsistency for a particular pair of literals to be inferred, given another particular condition. E.g., the following specifies that it is a contradiction to conclude two different values of the ordering-lead-time for the same order. The CLP KR's semantics enforce that the set of sanctioned conclusions respects (i.e., is consistent with) all the mutex's within the given CLP.

$\perp \; \leftarrow \; orderModificationNotice(?Order, ?X) \wedge$
$\quad\quad orderModificationNotice(?Order, ?Y)$
$\quad\quad\quad \mid \; (?X \neq ?Y).$

Next, we can extend the example to include actions and queries that are performed by procedural attachments – utilizing the *situated* extension of (courteous) logic programs. Space limitations prevent us from giving more than a few details on the situated aspects; see the author's website for more.

The following *effector statement*

$shouldInformCustomer(?X, ?Y)$
$\quad\quad :: e :: orderMgmt.request.mods.ack(?X, ?Y).$

associates a pure-belief predicate, i.e., $shouldInformCustomer$, (which is mentioned in some rules, here omitted due to space constraints) with an external procedure (here, a Java method), i.e., $orderMgmt.request.mods.ack$. During rule inferencing/execution, when a conclusion is drawn about the predicate, e.g., $shouldInformCustomer(request1049, denied)$, then the external procedure is invoked as a side-effectful action, e.g., the method $orderMgmt.request.mods.ack$ is called with its parameters instantiated to $(request1049, denied)$.

The following *sensor statement*

$$receiptDate(?X, ?Y)$$
$$:: s :: orderMgmt.request.getReceiptDate(?X, ?Y).$$

associates a pure-belief predicate, i.e., $receiptDate$ (of a request to modify an order), with an external procedure (here a Java method), i.e., $orderMgmt.request.getReceiptDate$. During rule inferencing/execution, when a rule antecedent condition (i.e., a literal in the rule's "if" part) is tested, e.g., $receiptDate(?Request, ?Day2)$, the external procedure is queried to provide information about that condition's truth (more precisely, for its answer bindings).

Unfortunately, space limitations in this version also prevent including the description of RuleML's XML syntax (e.g., for the leadtime example); see the author's website for details. The extant DTD version of the XML syntax is currently being extended to versions using XML Schema and RDF.

4 Related, Current, and Future Work

In current work, we are exploring how to provide a Rule mechanism to (other) emerging W3C[14] standards, especially: Semantic Web, notably its ontologies[15] aspect and RDF; XML Query for XML database querying; and P3P's APPEL rule language for user privacy policies. Each of these areas has an associated significant body and community of related fundamental research.

One direction for future work is pilot applications in e-business, including: contracting and negotiation, e.g., where rules are useful to represent product/service/deal descriptions; search, selection, and composition of Web Services; and security & privacy, e.g., where rules are useful to represent authorization policies. Here we would welcome a community of researchers and practitioners trying out the RuleML approach, developing requirements, and providing a feedback loop to further improvements and extensions.

References

[1] Chitta Baral and Michael Gelfond. Logic programming and knowledge representation. *Journal of Logic Programming*, 19,20:73–148, 1994. Includes extensive review of literature.

[2] Benjamin N. Grosof. Building Commercial Agents: An IBM Research Perspective (Invited Talk). In *Proc. 2nd International Conference and Exhibition on Practical Applications of Intelligent Agents and Multi-Agent Technology (PAAM97)*, http://www.demon.co.uk./ar/PAAM97, April 1997. Held London, UK. Also available as IBM Research Report RC 20835 at http://www.research.ibm.com .

[3] Benjamin N. Grosof, Yannis Labrou, and Hoi Y. Chan. A Declarative Approach to Business Rules in Contracts: Courteous Logic Programs in XML. In Michael P. Wellman, editor, *Proc. 1st ACM Conference on Electronic Commerce (EC-99)*. ACM Press, 1999.

[14] http://www.w3.org
[15] structured vocabularies specified using logic-based KR