An Approach to using XML and a Rule-based Content Language with an Agent Communication Language

Benjamin N. Grosof[1] and Yannis Labrou[2]

[1] IBM T.J. Watson Research Center,
P.O. Box 704, Yorktown Heights, NY 10598, USA
grosof@us.ibm.com (alt.: grosof@cs.stanford.edu),
http://www.research.ibm.com/people/g/grosof/
[2] Electrical Engineering and Computer Science Department
University of Maryland, Baltimore County, Baltimore, MD, 21250, USA
jklabrou@cs.umbc.edu,
http://www.cs.umbc.edu/~jklabrou

Abstract. We argue for an XML encoding of FIPA Agent Communication Language (ACL), and give an alpha version of it, called Agent Communication Markup Language (ACML), which we have implemented. The XML approach facilitates: (a) developing/maintaining parsers, integrating with WWW-world software engineering, and (b) the enriching capability to (hyper-)link to ontologies and other extra information. The XML approach applies similarly to KQML as well. Motivated by the importance of the content language aspect of agent communication, we focus in particular on business rules as a form of content that is important in e-commerce applications such as bidding negotiations. A leading candidate content language for business rules is Knowledge Interchange Format (KIF), which is currently in the ANSI standards committee process. We observe several major practical shortcomings of KIF as a content language for business rules in e-commerce. We argue instead for a knowledge representation (KR) approach based on Courteous Logic Programs (CLP) that overcomes several of KIF's representational limitations, and argue for this CLP approach, e.g., for its logical nonmonotonicity and its computational practicality. CLP is a previous KR that expressively extends declarative ordinary logic programs cf. Prolog; it includes negation-as-failure plus prioritized conflict handling. We argue for an XML encoding of business rules content, and give an alpha version of it, called Business Rules Markup Language (BRML), which we have implemented. BRML can express both CLP and a subset of KIF (i.e., of first-order logic) that overlaps with CLP. BRML expressively both extends and complements KIF. The overall advantages of an XML approach to content language are similar to those for the XML approach to ACL, and indeed complements the latter since content is carried within ACL messages. We have implemented both ACML and BRML/CLP; a free alpha prototype of BRML/CLP, called IBM CommonRules, was released on the Web in July of 1999.

1 Introduction

The concept of an Agent Communication Language (ACL) has its origins in the work of the Knowledge Sharing Effort (KSE). The KSE work gave birth to Knowledge Query and Manipulation Language (KQML) in the early 1990's, which in turn infuenced the Foundation for Intelligent Physical Agents (FIPA standards body) ACL[1]. (Terminology: In this paper, by "ACL" we mean either KQML (which has now several variants) or FIPA ACL. Since then the problem of an adequate semantics of an ACL has dominated the debate on ACL's. Despite the substantial amount of work on this problem, the issue of an agent's conformance with the ACL semantics is as thorny as ever [20] and moreover puts into question the degree of usefulness of semantic accounts. But even worse, the emphasis on ACL semantics has drawn attention away from other issues that are perhaps even more important to the success of ACL's: (1) how do agents find one another and manage to establish a "conversation"; (2) having achieved that, what is the "content" about which they actually talk; and (3) the relationship between ACL's and WWW technologies. We are interested in the latter two issues. [2]

KQML and FIPA ACL have evolved at a considerable distance from the mainstream of Internet technologies and standards. No Internet standardization organization has ACL's in their agenda. With the exception of the Artimis project (France Telecom), no major industry player has committed major resources to depend upon, or to develop, ACL's, although there are some plans for future work that will take advantage of FIPA technologies, as they become available. At the same time the WWW is a huge repository of information and agents are almost always referred to in conjuction with the WWW. ACL's are driving a great part of the agent work (FIPA ACL is the centerpiece of the FIPA effort); it is thus reasonable to suggest that ACL work ought to integrate easily with the WWW and to be able to leverage WWW tools and infrastructure. This motivates us to give (and to advocate) an Extensible Markup Language (XML) encoding of ACL messages, as a first step towards this kind of integration.

Agents, while conversing, exchange their information content; specifically we focus on the language used to describe it, i.e., the *content language* in ACL terminology. An ACL message's content layer, which contains descriptions in the content language, is distinct from the *propositional-attitude* layer which contains the (speech act type of) primitive of the ACL message. (Terminology: In this paper, by an ACL communication "primitive", we mean what KQML calls a "performative" and what FIPA ACL calls a "communicative act".) The KSE developed the Knowledge Interchange Format (KIF) as a general-purpose content language. However, it is important for ACL's to support multiple, e.g., special-purpose, content languages. We are particularly interested in representing business rules for e-commerce applications of communicating agents. For this purpose, we observe that KIF has significant shortcomings, notably, its inability

[1] http://www.fipa.org

[2] We do not deal with the first issue in this paper. See [13] for such a discussion.

to represent logical non-monotonicity. Accordingly, we give a new content language for business rules: an extended form of logic programs, with deep declarative semantics, encoded moreover in XML. This language, called Business Rules Markup Language (BRML), overcomes several limitations of KIF, yet broadly overlaps with KIF both syntactically and semantically; it thus extends and complements KIF.

Next, we give an outline of the remainder of this paper. In Section 2, we argue for the advantages of encoding ACL messages in XML and then present ACML, an XML language for that purpose. In Section 3 we review the content language concept and some existing content languages, then discuss our focus on business rules for e-commerce applications such as bidding negotiations. In Section 4, we review KIF and critique its shortcomings as a representation for business rules. In Section 5, we give a business rules content language, called *Courteous Logic Programs (CLP)*, that extends and complements KIF, while addressing several of KIF's shortcomings. In Section 6 we present BRML, the XML encoding of CLP. In Section 7, we describe our implementation. A free alpha prototype of BRML/CLP, called IBM *CommonRules*, was released on the Web in July of 1999. Current and future work directions are discussed in appropriate spots throughout the paper but we summarize them in Section 7.

2 XML Embodiment of FIPA ACL

In this section, we give an encoding of FIPA ACL messages in XML, and observe that using XML has several advantages. This leads us to suggest that in future industry practice, the preferred encoding for ACL messages should be XML rather than pure ASCII. (We are focusing on FIPA ACL but the same arguments and approach would apply to KQML too.) Finin, Labrou, and Grosof together first advocated this idea to FIPA during the FIPA meeting in Dublin, in July 1998. Although other groups of researchers have been considering a XML encoding for FIPA ACL, this paper is (to the the best of our knowledge) the first published treatment of this issue. As we will detail in Section 6, we advocate using XML also for the content of the ACL message itself, for similar reasons. Keep in mind, however, that the content need not be in XML even if the ACL message is in XML, or vice versa.

2.1 Brief Review of XML

XML is a language for creating markup languages that describe data. XML is a machine-readable and application-independent encoding of a "document", e.g., of a FIPA ACL message including its content.

In contrast to HTML which describes document structure and visual presentation, XML describes data in a human-readable format with no indication of how the data is to be displayed. It is a database-neutral and device-neutral format; data marked up in XML can be targeted to different devices

using, for example, eXtensible Style Language (XSL). The XML source by itself is not primarily intended directly for human viewing, though it is human-understandable. Rather, the XML is rendered using standard available XML-world tools, then browsed, e.g., using standard Web browsers or specialized other browsers/editors. (Netscape and Microsoft already are supporting XML in the latest versions of their Web browsers, for example.) One leading method for rendering is via XSL, in which one specifies a stylesheet.

XML is a meta- language used to define other domain- or industry-specific languages. To construct a XML language (also called a "vocabulary"), one supplies a specific *Document Type Definition (DTD)*, which is essentially a context-free grammar like the Extended BNF (Backus Naur Form) used to describe computer languages. In other words, a DTD provides the rules that define the elements and structure of the new language. For example, if we want to describe employee records, we would define a DTD which states that the <NAME> element consists of three other elements called <FIRST>, <MIDDLE>, and <LAST>, in that order. The DTD would also indicate if any of the nested elements is optional, can be repeated, and/or has a default value. Any browser (or application) having an XML parser could interpret the employee document instance by "learning" the rules defined by the DTD.

2.2 Review of ACL

The core semantics of an ACL is defined as the "*deep*" semantics (i.e., semantics in the sense of declarative knowledge-representation) of its (communication) primitives. This semantics are expressed in some knowledge representation language: SL in the case of FIPA ACL. This semantics only takes into account the *speaker*, the *hearer* (in *speech act* terminology) and the content of the communicative act. The speaker, the hearer and the content correspond to the `:sender`, the `:receiver` and the `:content` of the *syntactic* representation of the ACL. The previous canonical syntactic form of the ACL message (for both KQML and FIPA ACL) is a Lisp-like ASCII sequence.

The (previous) canonical ACL message syntax (both in FIPA ACL and KQML) further includes additional message parameters whose semantics go beyond that of the primitives. These parameters are unaccounted for in the deep semantics but are essential to the processing of an ACL message. In other words, the ACL includes several "*pragmatic*" (i.e., operational) aspects, in addition to the primitives aspect. One pragmatic aspect is parsing in and out of the ACL, i.e., digesting and composing well-formed ACL syntax (which is Lisp-like) to extract or insert message parameters. A second pragmatic aspect is queueing (and de-queueing) ACL messages for delivery through TCP or some other network protocol.

Further pragmatic issues being dealt with in the context of ACL efforts include the agent naming scheme, and the conventions for finding agents and initiating interaction; although, in our view, these issues are actually outside of the ACL's scope. Actually, the various APIs for KQML and FIPA ACL provide

nothing (as expected) regarding the actual processing of ACL messages (depending on the primitive), since respecting the deep semantics of the primitives is the responsibility of the application that makes use of those API's. Such API's today mainly take care of the parsing and queueing tasks mentioned above. Performing these tasks is what using KQML (or FIPA ACL, for that matter) has come to mean. For all intents and purposes, compliance with the ACL's specification means compliance with all these pragmatic conventions. Such conventions are not part of the standard (to the extent that the ACL semantics is standardized) and the subtle (or not so subtle) discrepancies amongst their implementations account in large part for the situation today in which there is often a lack of interoperability between systems using the same ACL. [3]

2.3 Introducing ACML

Next, we give an alpha-version specification of FIPA ACL in XML, which we call **Agent Communication Markup Language (ACML)**. To begin with, we need to define a DTD for ACML [4]. We have indeed defined an alpha-version DTD for ACML, and have a running prototype implementation of ACML that uses this DTD.

We begin with an example of a XML encoding of a FIPA ACL message. Figure 1 shows an example FIPA ACL message, in the previous (ASCII, Lisp-like) syntax. Figure 2 shows the same FIPA ACL message encoded in XML, i.e., in ACML. The content is a KIF expression which is not encoded in XML in this example. The DTD for ACML is shown in Figure 3. This is an alpha version.

The deep semantics of the communication primitives in ACML is simply taken to be the same as previously. This semantics is not affected by encoding in XML instead of the previous ASCII; it is defined independently of the choice of syntactic encoding.

By XML-ifying the syntactic representation we enhance (i.e., extend) the (previous) canonical (pure ASCII) syntactic representation by introducing markup for parsing (the "`tags`", in XML terminology). This markup significantly facilitates the development effort needed for parsing in and out.

The XML representation also facilitates introducing pragmatic/operational elements that go beyond what the pure ASCII previous syntax did: notably, via *links* (in a similar sense as does HTML compared to ASCII). And we indeed introduced such extras in our alpha DTD and example. For example, the ACL message of Figure 2 includes information beyond what is equivalent to that in Figure 1. Here (Figure 2), the receiver is not just some symbolic name but is also a URL that points to a particular network location which could provide additional information about the receiver agent's identity (e.g., how to contact its owner, its network ports, etc.).

[3] The differences in sets of primitives used and their intended meaning constitute a second-in-order interoperability barrier that is not confronted due to these more mundane "lower-level" obstacles.

[4] The same will be done for the content language (see Section 6).

2.4 Advantages of XML Approach

Encoding ACL messages in XML offers some advantages that we believe are potentially quite significant.

(1) The XML-encoding is **easier to develop parsers for** than the Lisp-like encoding. The XML markup provides parsing information more directly. One can use the off-the-shelf tools for parsing XML — of which there are several competent, easy-to-use ones already available — instead of writing customized parsers to parse the ACL messages. A change or an enhancement of the ACL syntax does not have to result to a re-writing of the parser. As long as such changes are reflected in the ACL DTD, the XML parser will still be able to handle the XML-encoded ACL message. In short, a significant advantage is that the process of *developing or maintaining* a parser is much simplified.

Indeed, we have first-hand experience that this parsing advantage is significant. In our own implementation efforts, we have developed parsers for FIPA ACL and for content languages (both KIF and logic programs), both for ASCII encoding and for XML encoding.

(2) More generally, XML-ifying makes ACL more "**WWW-friendly**", which **facilitates Software Engineering** of agents. Agent development ought to take advantage and build on what the WWW has to offer as a software development environment. XML parsing technology is only one example. Using XML will facilitate the practical integration with a variety of Web technologies. For example, an issue that has been raised in the ACL community [5] is that of addressing security issues, *e.g.* authentication of agents' identities and encryption of ACL messages, at the ACL layer. The WWW solution is to use certificates and SSL. Using the same approach for agent security considerations seems much simpler and more intuitive than further overloading ACL messages and the ACL infrastructure to accommodate such a task.

As we mentioned earlier, the operational semantics of the pragmatic aspects of ACL can differ subtly between implementations or usages, and there is today a practical problem of interoperability. XML can help with these pragmatics, by riding on standard WWW-world technologies: to facilitate the engineering, and as a by-product to help standardize the operational semantics, thereby helping make interoperability really happen.

(3) Because XML incorporates links into the ACL message, this takes a significant step toward addressing the problem (or representational layer) of specifying and sharing the **ontologies** used in an ACL message's content. The values of the ACL parameters are not tokens anymore, but links that can point to objects and/or definitions. Although the `ontology` slot has been present since the inception of ACLs, the ACL community has not been very clear on how this information is to be used by the agent. This vagueness, further compounded by the scarcity of published ontologies, can be addressed by "interfacing" the ACL message to the knowledge repository that is the WWW.

(4) More generally, **links may be useful for a variety of other purposes**. For example, the `receiver` parameter might have a link to network location that

[5] Private communication at FIPA meetings

provides information about the **agent's identity**: e.g., its owner, contact and administrative information, communication primitives that the agent understands, network protocols and ports at which it can receive messages, conversation protocols it understands, *etc.*. This type of information is necessary for a establishing an extended interaction with another agent and has to somehow be available to an agent's potential interlocutors. The same argument can be made about the other message parameters.

3 ACL Content Languages, e.g., for Business Rules

3.1 Layered Approach of Knowledge Sharing Effort

Our and many other current efforts in inter-agent communication approaches are influenced by the pioneering approach of the Knowledge Sharing Effort [15, 16] (KSE)[6] The KSE was initiated as a research effort circa 1990 with encouragement and relatively modest funding from U.S. government agencies (DARPA especially). The KSE was highly active for roughly five years thereafter, and enjoyed the participation of dozens of researchers from both academia and industry. Its goal was to develop techniques, methodologies and software tools for *knowledge sharing and knowledge reuse* between *knowledge-based (software) systems*, at *design*, *implementation*, or *execution* time. Agents, especially intelligent agents, are an important kind of such knowledge-based systems (other kinds include expert systems or databases, for example). The central concept of the KSE was that knowledge sharing requires communication, which in turn, requires a common language; the KSE focused on defining that common language.

In the KSE model, agents (or, more generally, knowledge-based systems) are viewed as (virtual) knowledge bases that exchange propositions using a language that expresses various *propositional attitudes*. Propositional attitudes are three-part relationships between: (1) an agent, (2) a content-bearing proposition (*e.g.*, "it is raining"), and (3) a finite set of propositional attitudes an agent might have with respect to the proposition (*e.g.*, believing, asserting, fearing, wondering, hoping, *etc.*). For example, $< a, fear, raining(t_{now}) >$.

The KSE model includes three **layers** of representation: (1) specifying propositional attitudes; (2) specifying propositions (i.e., "knowledge") — this is often called the (propositional) **content** layer; and (3) specifying the *ontology* [12] (i.e., vocabulary) of those propositions. The KSE accordingly includes a component (with associated language) for each of these: Knowledge Query and Manipulation Language (KQML) for propositional attitudes, **Knowledge Interchange Format (KIF)** [4][7] for propositions, and Ontolingua [3] (which had supporting software tools) for ontology.

Within the KSE approach, the three representational layers are viewed as mainly independent of another. In particular, the language for propositional

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

content (i.e., the *content language*) can be chosen independently from the language for propositional attitudes. In other words, in the KSE approach, the role of an ACL such as FIPA's is only to capture propositional attitudes, regardless of how propositions are expressed, even though propositions are what agents will be "talking" about.

In a similar spirit, the approach of the technical committee that worked on FIPA ACL is that the content language should be viewed as orthogonal to the rest of the ACL message type.

The KSE focused especially on developing one general-purpose content language: KIF. However, the KSE also recognized that it is important to support multiple special-purpose content languages, since some are more expressive or more convenient for a particular purpose. Indeed, the KSE also included a fourth component effort (abbreviated "KRSS") devoted to defining a special-purpose content language for "description logics" (a.k.a. "terminological logics", descended from KL-ONE).

We agree with the view that it is important to support multiple content languages. Beyond the KSE, a number of important specialized content languages have been developed which are particularly good at describing certain fields. For example, STEP (Standard for the Exchange of Product Model Data) [12] is an ISO standards project working towards developing mechanisms for the representation and exchange of a computerized model of a product in a neutral form. SGML is an example of a language, which is designed to describe the logical structure of a document. There are special languages for describing workflow, processes, chemical reactions, etc. SQL and OQL are somewhat more general content languages: for relational and object databases.

3.2 Business Rules in E-Commerce as focus

Motivated by the importance of the content language aspect of agent communication, we focus in particular on rules as a form of content that is important in e-commerce applications such as bidding negotiations, i.e., "business rules". We are particularly interested in this kind of application, and have been developing techniques for it [10] [17] (to describe these is beyond the scope of this paper, however).

In bidding negotations, agents exchange requests for bids, (i.e., proposals), make proposals, make counter-proposals, until agreeing or giving up. Rules are useful to represent the contents of these proposals and requests for proposals: e.g., to describe the products/services, prices, quantities, delivery dates, customer service agreements, contractual terms & conditions, and other surrounding agreements that together constitute the content of a bid. Rules are also useful to represent relevant aspects of business processes, e.g., how to place an order, respond to an RFQ, return an item or cancel a delivery.

The usefulness of rules for the overall area of agent communication, particularly for such e-commerce applications is based largely on their following advantages relative to other software specification approaches and programming

languages. First, rules are at a relatively high level of abstraction, closer to human understandability, especially by business domain experts who are typically non-programmers. Second, rules are relatively easy to modify dynamically and by non-programmers.

Rules provide an expressive yet automatically executable form for the substance of these specifications. Rules with deep declarative semantics[8] are valuable because they help enable business rules to be specified dynamically, i.e., at runtime, and relatively easily by business domain experts who are non-programmers.

There are a number of different rule representations in wide deployment today. A major challenge in communicating content between e-commerce agents is thus the heterogeneity of rule representations (within agents/applications) to be integrated, e.g., during negotation. In translating content via a common rule representation, deep semantics (in the sense of declarative KR) is desirable. However, one can only hope to obtain deep semantics for expressive *cores*, i.e., for the expressive cases that overlap between the source and target rule KR's. Beyond the cores, translation must be performed with superficial semantics.

To begin with, we are focusing on three broad families of rule representations that are currently commercially important for business rules in e-commerce. These are both executable and practically important in the software world at large. One family is logic programs (LP's): including, but not limited to, Prolog. Logic programs have a general, declarative sense; they can be forward-chaining as well as backward-chaining, and need not be a general-purpose programming language in the manner of Prolog. Baral & Gelfond [1] gives a useful review of declarative logic programs as a KR. Another family is production rules: descendants of OPS5 [2], e.g., the public domain system Jess[9]. A third (relatively loose) family is Event-Condition-Action (ECA) rules. Both logic programs and ECA rules are important in commercial databases[19] [18] and related standards (including SQL). Rules in these three families are to be found, for example, in object-oriented applications and worfklows, as well.

4 KIF and its Shortcomings for Business Rules Content

A leading candidate content language for rules is KIF. KIF is currently well along in the ANSI standards committee process. Supporting or endorsing KIF is also being considered informally in several other standards efforts relevant to agent communication, e.g., FIPA.

KIF has pioneered the concept of a KR content language for agent communication. That said, there are some important differences between (1) the goals of the KIF effort and (2) our goals for a business rules content language (for practical e-commerce agents' communication). The KIF effort's goals were initially

[8] in the sense of declarative knowledge representation, in which a set of premises entails a set of conclusions, independent of the inferencing procedure, e.g., whether it is forward or backward direction, what its control strategy is, *etc.*.

[9] http://herzberg.ca.sandia.gov/jess/ . Jess is written in Java and is an update of CLIPS (http://www.ghg.net/clips/CLIPS.html).

to facilitate exchange among research systems rather than commercial systems. Also, it aimed to help at least somewhat with exchange of many forms of knowledge beyond just rules. It was designed with an orientation towards knowledge as a non-executable specification as much or more than towards knowledge as executable. Finally, the KIF effort has focused more on a highly inclusively expressive representation than on ease of developing translators in and out of that representation.

KIF is a prefix[10] version of first-order predicate calculus (i.e., first-order classical logic) with extensions to support the "quote" operator (thus enabling additional expressiveness akin to that of classical higher-order logic) and definitions. The language description includes a specification not only for its syntax but also for its semantics. Its deep semantics is based on classical logic, which is logically monotonic. Its primary focus (in terms of deep semantics) is on first-order logic, which is highly expressive and computationally intractable for the general case (as well as logically monotonic).

KIF can express a broad class of rules. However, it has several important shortcomings as a content language for business rules in e-commerce. In particular, it has two shortcomings of its fundamental knowledge representation.

(1) KIF is a *logically monotonic* KR. KIF cannot conveniently express rules that are **logically non-monotonic**, e.g., rules that employ **negation-as-failure** or **default rules**. Thus it cannot conveniently express **conflict handling**, e.g., where some rules are subject to **override by higher-priority conflicting rules**, e.g., by special-case **exceptions**, by more-recent **updates**, or by higher-authority sources. Most commercially important rule systems employ non-monotonic reasoning as an essential, highly-used feature. Typically, they employ some form of negation-as-failure. Often they employ some form of prioritized override between rules, e.g., the static rule sequence in Prolog or the computed rule-activation sequence/"agenda" in OPS5-heritage production rule systems.

Early in the KIF effort, incorporating logical non-monotonicity was considered seriously. However, no technical agreement could be reached on an approach, largely because of its ambitions for great expressive generality in the direction of full classical logic. The current ANSI draft proposal of KIF is logically monotonic.

(2) KIF is a *pure-belief* KR. KIF cannot conveniently express "**procedural attachments**": the association of procedure calls (e.g., a call to a Java method ProcurementAuthorization.setApprovalLevel) with belief expressions (e.g., a logical predicate such as approvalAuthorizationLevel). Procedural attachments are crucial in order for rules to have actual effect beyond pure-belief inferencing, i.e., for actions to be invoked/performed as a result after rule conclusions are inferred. While procedures can of course be invoked by an application based on KIF premises or conclusions, KIF provides no way to express this, and its se-

[10] The current draft ANSI specification of KIF (http://logic.stanford.edu/kif/dpans.html) also includes an infix version of KIF intended for human consumption rather than automated exchange.

mantics do not treat the connection to such invocations, i.e., to such procedural attachments.

5 A Logic Program Based Business Rule Content KR

5.1 Overall Approach: Ordinary, Courteous, and Situated LP's

We identified two fundamental shortcomings of KIF as a KR for business rules content: logical non-monotonicity and procedural attachments. In this paper, we focus on selecting a business rules content KR to remedy the first shortcoming only. We select a business rules content KR to enable logical non-monotonicity, including two steps. (1) Negation-as-failure, a basic form of non-monotonicity, is the first step. (2) Prioritized override between conflicting rules (i.e., prioritized default rules and conflict handling) is the second step.

Our approach is to use ordinary Logic Programs to provide the first step. By Logic Program, we mean in the declarative sense, e.g., cf. [1][11]. Inferencing for LP's can be run forward or backward, using a variety of control strategies and algorithms; Prolog, by contrast, does backward-only inferencing, using a particular control strategy. Ordinary LP's (OLP's) offer several other significant advantages beyond enabling non-monotonicity, including: computational tractability, wide practical deployment, semantics shared with other practically important rule systems, relative algorithmic simplicity, yet considerable expressive power.

Our approach is then to use **Courteous Logic Programs (CLP's)**, an expressive extension of ordinary Logic Programs, to provide the second step. Courteous Logic Programs [6] [9] [7] provide a computationally low-overhead, semantically-clean capability for prioritized handling of conflicts between rules. CLP's permit classical negation; syntactically they also permit optional rule labels which are used as handles for specifying prioritization.

In current work, we are also enabling procedural attachments as well — in a semantically clean manner (i.e., declaratively in a particular well-defined sense). Our approach to enabling procedural attachments is based on **Situated Logic Programs**, another expressive extension of ordinary logic programs. Situated Logic Programs [5] [11] hook beliefs to drive procedural API's. Procedural attachments for testing conditions (sensing) and performing actions (effecting) are specified as part of the knowledge representation: via sensor and effector link statements. Each sensor or effector link associates a predicate with an attached procedure.[12]

5.2 Ordinary LP's: Core & Advantages

Our point of departure for the business rules content KR is pure-belief ordinary LP's. "Pure-belief" here means without procedural attachments.

[11] They call an ordinary LP: a "general" LP. This is also known in the literature as a "normal" LP, and also sometimes as (declarative) pure Prolog.

[12] Note that "link" here does not mean in the sense of an XML or HTML hypertext link.

OLP's include negation-as-failure and thus support basic non-monotonicity. Yet they are relatively simple, and are not overkill representationally. OLP's are also relatively fast computationally. Under commonly-met restrictions (e.g., no logical functions of non-zero arity, a bounded number of logical variables per rule), inferencing (i.e., rule-set execution) in LP's can be computed in (worst-case) polynomial-time. By contrast, under similar restrictions, first-order-logic (cf. KIF) inferencing is (co-)NP-hard.

To obtain deep semantics that is/will-be shared widely among heterogeneous rule systems, however, the core must be an expressively restricted case of OLP's. Our alpha-version choice of this expressive restricion is: "predicate-acyclic" (pure-belief) OLP's — below, we discuss this in more detail. This core has a deep semantics that is useful, well-understood theoretically and highly declarative. Moreover, this semantics reflects a consensus in the rules representation community beyond just the LP community: this semantics is widely shared among all three of the rule system families we mentioned in subsection 3.2.

This core is also relatively computationally efficient, in the sense we described above.

The unrestricted case of declarative OLP's, with unrestricted recursion/cyclicity (in a sense explained below) interacting with negation-as-failure, has problems semantically, is more complex computationally and, perhaps even more importantly, is more difficult in terms of software engineering. It requires more complicated algorithms and is not widely deployed.

OLP's have been widely deployed practically, in contrast to full first-order-logic which has not been. Moreover, there is a large population of software developers who are familiar with Prolog and OLP's, in contrast to general first-order-logic theorem-proving for which there is not.

5.3 Ordinary LP's: Semantics & Recursion

Ordinary LP's have been well-studied, and have a large literature (reviewed, for example, in [1]). For several broad but restricted expressive cases, their (declarative) semantics is uncontroversial.[13] However, OLP's have problematic semantics for the unrestricted case, due essentially to the interaction of recursion with negation-as-failure. "Recursion" here means that there is a *cyclic* (path of syntactic) dependency among the predicates (or, more generally, among the ground atoms) through rules. [14]

There is a lack of consensus in the research community about which semantics to adopt for the fully general case of OLP's: e.g., well-founded semantics versus

[13] e.g., for the predicate-acyclic, stratified, locally stratified, and weakly stratified cases; these form a series of increasing expressive generality

[14] In each rule, the predicate(s) appearing in the consequent/head of the rule has a directed dependency arc to each of the predicates appearing in the antecedent/body of the rule. Accumulating such dependency arcs for a whole rule set, and taking their transitively closed paths, defines which predicates are dependent on which others for a given LP.

stable semantics, etc.; these semantics coincide for the uncontroversial restricted cases but diverge beyond that. Under the well-founded semantics, probably the currently most popular semantics, the unrestricted case is tractable.

Our approach for an initial practically-oriented LP-based business rules content KR is to keep to expressively restricted cases that have uncontroversial (i.e., consensus) semantics; these have other virtues as well: e.g., they are algorithmically and computationally simpler. More precisely, our approach is to define/expect deep semantics (including for translation between agents) only for these restricted cases.

Our starting choice for such an expressive restriction is: **predicate-acyclic**, i.e., where there are no cycles of (syntactic) dependency among predicates. This expressive restriction can be checked syntactically with a relatively simple algorithm and with relatively low computational cost. Inferencing for the predicate-acyclic case is also simpler algorithmically and computationally than for the expressively unrestricted case.

In our XML embodiment (next section) of the LP-based content language, we define an alpha-version DTD that is syntactically inclusive: it permits unrestricted OLP's. It is thus useful there to have an optional tag to indicate which semantical variant of LP's is intended: the DTD accordingly defines an optional "documentation" link which can be used to specify the intended semantics (e.g., well-founded versus stable). For the alpha-version, our approach is to choose the well-founded semantics to be the default semantics for the expressively unrestricted case.

5.4 Courteous Logic Programs

Courteous LP's expressively generalize OLP's by adding the capability to conveniently express prioritized conflict handling, i.e., where some rules are subject to override by higher-priority conflicting rules. For example, some rules may be overridden by other rules that are special-case exceptions, more-recent updates, or from higher-authority sources. Courteous LP's facilitate specifying sets of rules by merging and updating and accumulation, in a style closer (than ordinary LP's) to natural language descriptions.

Courteous LP's also expressively generalize ordinary LP's and permit **classical negation** to appear in head (i.e., consequent) or body (i.e., antecedent) literals (negation-as-failure must appear outside, not inside, the scope of classical-negation). They also permit rules to have optional labels, which are used as handles for specifying priorities. A syntactically-reserved (but otherwise ordinary) predicate *overrides* is used to specify prioritization. Priorities are represented via a fact comparing rule labels: $overrides(lab1, lab2)$ means semantically that a rule having label $lab1$ has higher priority than another rule having label $lab2$. If two such rules conflict, then the rule with the higher priority will win the conflict; the lower priority rule's head will not be concluded.

The prioritization specified is a partial ordering, rather than a total ordering. Classical negation is enforced: p and classical-negation-of-p are never both concluded, for any belief expression p.

In the alpha-version business rules content KR (BRML) given here, the Courteous LP KR is also expressively restricted in two further regards cf. [6]: (1) priority is specified via ground facts only, and (2) priority is specified to be a strict partial order. Elsewhere [7] [9] [8] [10], we give an expressively generalized version of Courteous LP's and BRML that relaxes these restrictions and the predicate-acyclicity restriction, and adds several further expressive generalizations, notably to permit conditional pairwise mutual exclusion constraints (*mutex*'s) that specify the scope of conflict handling.

Courteous LP's have several virtues semantically and computationally. A Courteous LP is guaranteed to have a **consistent**, as well as unique, set of conclusions. Priorities and merging behave in an intuitively natural fashion. Execution (inferencing) of courteous LP's is **fast**: only relatively low computational overhead is imposed by the conflict handling.

From a software engineering viewpoint as well, CLP's are a relatively straightforward extension of OLP's. A CLP can always be **tractably compiled into a semantically equivalent OLP** — indeed, we have implemented CLP's using such a "*courteous compiler*" [7] [9] [8] [10].

Detailed computational complexity analysis for courteous LP inferencing and the courteous compiler is given in [6] [7] [8] [10]; next, we summarize that analysis. The complexity of courteous compilation is worst-case cubic, both in time and in output size. Suppose the input LP, having size n, is either ground or Datalog (no logical functions of more than zero arity), and has an upper bound v on the number of logical variables appearing in any rule. As we mentioned earlier, the worst-case time complexity of inferencing in OLP's under these restrictions is tractable (i.e., polynomial). Courteous LP inferencing then has the same worst-case time and space complexity as: OLP inferencing where the bound v on the number of variables per rule has been increased to $v + 2$.

There are several other formalisms for prioritized LP's that have similar syntax to Courteous LP's but different semantics in regard to conflict handling (see [6] [7] for a review). A direction in our current work is to explore this dimension of heterogeneity.

5.5 Relationship to KIF; Discussion

In this subsection, we discuss how the alpha-version business rules content KR, i.e., CLP cf. [6] encoded in XML as BRML, relates to KIF.

Syntactically, the alpha CLP adds two (optional) features to OLP: classical negation and rule labels. KIF permits classical negation but not negation-as-failure. Also KIF remarkably lacks rule labels (or rule names/id's), even though this is rather routine as a basic naming/scoping mechanism in rule specification systems and many programming languages. Syntactically, the alpha CLP thus adds two (optional) features to KIF: negation-as-failure and rule labels.

Syntactically, OLP and first-order-logic/KIF overlap to a considerable degree: OLP without negation-as-failure is logically monotonic[15]. Syntactically and se-

[15] when one interprets lack of membership in the minimal/least model of the OLP as corresponding to classical non-entailment rather than to classical falsity

mantically, such monotonic OLP is simply Horn and is thus a restricted case of first-order logic/KIF. Semantically, OLP entailment/inferencing is sound but incomplete when compared to first-order-logic (FOL). The incompleteness can be described as: an OLP's entailed conclusions are equivalent to a set of ground atoms.

Syntactically, CLP and FOL/KIF overlap to an even more considerable degree: CLP without negation-as-failure is logically monotonic. Such monotonic CLP with its labels omitted or ignored is thus syntactically a restricted case of FOL/KIF. Semantically, a monotonic CLP may contain conflict; we say it is "classically consistent" or "conflict free" when it is consistent when viewed as FOL. Semantically, a consistent monotonic CLP is sound but incomplete when compared to FOL. The incompleteness is similar to that of OLP; it can be described as: a CLP's entailed conclusions are equivalent to a set of ground classical literals.

6 XML Embodiment: Business Rules Markup Language

Just as we have defined an XML encoding for ACL messages in Section 2.3, we have defined an XML encoding for CLP rulesets. We refer to this language as **Business Rules Markup Language (BRML)**. BRML inherits the deep semantics of CLP.

Figure 4 gives an example of a single-rule CLP ruleset, in BRML. See the IBM CommonRules release package (`http://alphaworks.ibm.com`) for an extensive collection of examples, including about price discounting, targeted ads/promotions, refunds, creditworthiness, etc.. Figure 5 gives the (alpha) BRML DTD. The XML encoding extends the pure ASCII syntactic representation of CLP (not shown here for reasons of space and focus) with parsing information (and eventually with various optional links). The optional `documentation` attribute in the BRML DTD could point to a link which has information such as the semantical variant of the language.

In the draft DTD shown, we do not yet allow a predicate (or another token such as a logical constant or function, *etc.*) to have an associated link, because here we are focused on specifying the basic XML encoding of CLP. However, we plan to permit such links: e.g., the `loyalCustomer` predicate, for example, could then point to a URL containing a document that provides an account in natural language of what the particular company considers a loyal customer. Or, in the case of the example of Figure 2, the particular laptop for sale could include a linked picture and a URL with the full natural-language description of the laptop's technical specification.

The **advantages of an XML encoding** for business rules content are similar to those for ACL that we discussed in Section 2. As compared to plain ASCII text, XML is easier to automatically parse, generate, edit, and translate: because there are standard XML-world tools for these tasks. The hyper-text (i.e., links) aspects of XML are also useful. For example, a rule set may via XML have some associated URL's which point to documents describing that rule set's knowl-

edge representation or authors or application context. Or it may have associated URL's which point to tools for processing that rule set, e.g., to execute it, edit it, analyze it, or validate it (syntactically or semantically). Particularly useful for our nearer-term purposes is that an associated URL may point to documents describing the semantics and algorithms for translator services or components, as well as to translator tools and examples. Representing business rules in XML has a further advantage: it will complement domain-specific ontologies (i.e., vocabularies) available in XML. Many such ontologies exist already, and many more are expected to be developed in the next few years, including in e-commerce domains.

Further discussion of our DTD: Actually, our BRML DTD permits a syntactic superset of our alpha expressive core, i.e. a superset of CLP cf. [6]. Applications using the BRML need to perform additional "*validation*", i.e., checking of syntactic restrictions, beyond what is furnished by XML parsers that validate with respect to the DTD. However, such additional syntactic validation would be necessary even if the DTD was as "tight" as XML made possible; various other conditions such as predicate-acyclicity are impractically difficult (if not impossible) to capture in a DTD.

As a syntactic convenience, we permit the OR connective and nested subexpressions to appear in the body, and we permit the AND connective to appear in the head. This does not change the essential expressiveness of OLP or CLP (see, e.g., [14]) [16].

It appears fairly straightforward to extend our DTD in stages so as to express full first-order logic and then full KIF. A direction for future work is to create a DTD, maximally compatibly with BRML, that expresses full KIF.

7 Discussion: Implementation; Future Work Summary

We have a **running prototype implementation** of ACML, and of BRML and Courteous LP's as a Java library. Based on the DTD's we gave earlier, this includes encoding and parsing in/out in both XML and ASCII (including KIF for the content). It also includes translators to two other ASCII rule representations in the logic program family, used by previously existing OLP inferencing engines built by others and implemented in C. One is backward-direction: XSB, by David Warren *et al*, http://www.cs.sunysb.edu/~sbprolog/ . The other is exhaustive forward-direction: Smodels (first version), by Ilkka Niemela and Patrik Simons, http://saturn.hut.fi/html/staff/ilkka.html . All the encoding, parsing, and translating preserves the deep semantics of the alpha core that we described in Section 5. The implementation further includes a courteous compiler, and a rule inferencing/execution engine.

The prototype implementation of BRML and Courteous LP's, called **IBM CommonRules** was released free on the Web in July of 1999: at

[16] though in the worst-case depending on inferencing engine implementation this may cost exponential time/space caused by converting to the representation without OR's

`http://alphaworks.ibm.com`. As of Nov. 10 1999, there have been more than 2000 downloads of CommonRules. Overviews of CommonRules, with e-commerce examples, are given in [9] [10], and its courteous compiler algorithms are given in [7] [8]. The XML DTD's are under ongoing (non-radical) revision; see the authors' websites and CommonRules for updated versions.

Future work includes extending this XML content language expressively in multiple directions. One such direction is to cover full KIF; another is to incorporate semantically-clean procedural attachments, cf. the existing Situated Logic Programs KR; a third is to expressively generalize the Courteous LP conflict handling aspects .

Acknowledgements

Hoi Y. Chan (IBM), Michael Travers (IBM), and Xiaocheng Luan (of UMBC, while at IBM), contributed to the current implementation of the CLP KR, BRML, and the associated translators. Michael Travers' contributed especially to the XML embodiment, using his Skij tool. Hoi Y. Chan and Miao Jin (UMBC) contributed to the XML DTD's. Tim Finin (UMBC) contributed to the formulation of our ideas for the XML embodiment of the FIPA ACL, which he first presented at the FIPA meeting in Dublin, in July 1998.

References

1. Chitta Baral and Michael Gelfond. Logic programming and knowledge representation. *J. Logic Programming*, 19,20:73–148, 1994. Includes extensive review of literature.
2. Thomas Cooper and Nancy Wogrin. *Rule-based Programming with OPS5*. Morgan Kaufmann, San Francisco, CA, 1988. ISBN 0-934613-51-6.
3. Adam Farquhar, Richard Fikes, and James Rice. The ontolingua server: A tool for collaborative ontology construction. In *KAW96*, Nov. 1996.
4. M. Genesereth and R. Fikes et. al. Knowledge interchange format, version 3.0 reference manual. Technical report, Computer Science Dept., Stanford Univ., 1992.
5. Benjamin N. Grosof. Building Commercial Agents: An IBM Research Perspective (Invited Talk). In *Proc. 2nd Intnl. Conference and Exhibition on Practical Applications of Intelligent Agents and Multi-Agent Technology (PAAM97)*, Practical Application Company Ltd., P.O. Box 137, Blackpool, Lancashire, FY2 9UN, UK. http://www.demon.co.uk./ar/PAAM97, Apr. 1997. Held London, UK. Also available as IBM Research Report RC 20835.
6. Benjamin N. Grosof. Prioritized Conflict Handling for Logic Programs. In Jan Maluszynski, editor, *Logic Programming: Proceedings of the International Symposium (ILPS-97)*, MIT Press, Cambridge, MA, USA, 1997. http://www.ida.liu.se/~ilps97. Extended version available as IBM Research Report RC 20836.
7. Benjamin N. Grosof. Compiling Prioritized Default Rules Into Ordinary Logic Programs. IBM Research Report RC 21472, 1999.

8. Benjamin N. Grosof. A Courteous Compiler from Generalized Courteous Logic Programs To Ordinary Logic Programs (Preliminary Report). Technical report, IBM T.J. Watson Research Center, included as documentation in the IBM CommonRules 1.0 alpha prototype Web release of July 30, 1999 at http://alphaworks.ibm.com . This is a supplementary followon to IBM Research Report RC 21472. Revised version forthcoming as another IBM Research Report.
9. Benjamin N. Grosof. DIPLOMAT: Compiling Prioritized Default Rules Into Ordinary Logic Programs, for E-Commerce Applications (extended abstract of Intelligent Systems Demonstration). In *Proceedings of AAAI-99*, Morgan Kaufmann, San Francisco, CA, USA, 1999. Extended version available as IBM Research Report RC21473, May 1999.
10. 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, *Proceedings of 1st ACM Conference on E-Commerce (EC-99)*, New York, NY, USA, 1999. ACM Press. `http://www.ibm.com/iac/ec99/` or `http://www.acm.org`. Held Denver, CO, USA, Nov. 3–5, 1999. Extended version to be available as an IBM Research Report.
11. Benjamin N. Grosof, David W. Levine, Hoi Y. Chan, Colin P. Parris, and Joshua S. Auerbach. Reusable Architecture for Embedding Rule-Based Intelligence in Information Agents. In *Proceedings of the ACM Conference on Information and Knowledge Management (CIKM-95) Workshop on Intelligent Information Agents*, `http://www.cs.umbc.edu/iia/`, December 1995. Held Baltimore, MD. Paper also available as IBM Research Report RC 20305.
12. Thomas R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 2:199–220, 1993.
13. Yannis Labrou, Tim Finin, and Yun Peng. Agent communication languages: the current landscape. *IEEE Intelligent Systems*, May 1999.
14. J. W. Lloyd. *Foundations of Logic Programming, 2nd ed.*. Springer, Berlin, Germany, 1987.
15. R. Neches, R. Fikes, T. Finin, T. Gruber, R. Patil, T. Senator, and W. Swartout. Enabling technology for knowledge sharing. *AI Magazine*, 12(3):36–56, Fall, 1991.
16. Ramesh S. Patil, Richard E. Fikes, Peter F. Patel-Schneider, Don McKay, Tim Finin, Thomas Gruber, and Robert Neches. The DARPA knowledge sharing effort: Progress report. In Michael Huhns and Munindar Singh, editors, *Readings in Agents*. Morgan Kaufmann, 1997. (reprint of KR-92 paper).
17. Daniel M. Reeves, Benjamin N. Grosof, Michael Wellman, and Hoi Y. Chan. Toward a Declarative Language for Negotiating Executable Contracts. In *Proc. AAAI-99 Workshop on Artificial Intelligence in Electronic Commerce (AIEC-99)*, 1999. Proceedings published by AAAI/MIT Press (http://www.aaai.org) as Technical Report, also available at AIEC website `http://www.cs.umbc.edu/aiec/`. Also available as IBM Research Report RC 21476.
18. Jeffrey D. Ullman. *Principles of Database and Knowledge-Base Systems, volume 1*. Computer Science Press, Rockville, Maryland, 1988.
19. Jeffrey D. Ullman and Jennifer Widom. *A First Course in Database Systems*. Prentice-Hall, 1997.
20. Michael Wooldridge. Verifiable semantics for agent communication languages. In *International Conference on Multi-Agent Systems (ICMAS'98)*, Paris, France, 1998.

```
(inform
  :sender   jklabrou
  :receiver grosof
  :content  (CPU libretto50 pentium)
  :ontology laptop
  :language kif)
```

Fig. 1. A FIPA ACL message.

```
<?xml version="pre-1.0"?>
<!DOCTYPE fipa_acl SYSTEM "fipa_acl.dtd">
<message>
  <messagetype>
     inform
  </messagetype>
  <messageparameter>
     <sender link="http://www.cs.umbc.edu/~jklabrou">
     jklabrou
     </sender>
  </messageparameter>
  <messageparameter>
     <receiver link="http://www.research.ibm.com/people/g/grosof/">
     grosof
     </receiver>
  </messageparameter>
  <messageparameter>
     <ontology link="http://www.cs.umbc.edu/~jklabrou/
                                          ontology/laptop.html">
     laptop
     </ontology>
  </messageparameter>
  <messageparameter>
     <content>
     (CPU libretto50 pentium)
     </content>
  </messageparameter>
  <messageparameter>
     <language link="http://www.stanford.edu/kif.html">
     kif
     </language>
  </messageparameter>
</message>
```

Fig. 2. An example of a FIPA ACL message encoded in XML, i.e., expressed in ACML. Notice that the XML encoding carries additional information as compared to the canonical ASCII encoding: in particular, links (as well as parsing information).

```
<?xml version="pre-1.0" encoding="US-ASCII"?>

<!ENTITY % messagetp "accept-proposal | agree | cancel |cfp |
confirm | disconfirm | failure | inform | inform-if |
inform-ref | not-understood | propose | query-if | query-ref |
refuse | reject-proposal | request | request-when |
request-whenever | subscribe">

<!ELEMENT message (messagetype, messageparameter* )>
<!ELEMENT messagetype (%messagetp;)>


<!ELEMENT messageparameter (sender | receiver | content |
reply-with | reply-by| in-reply-to | envelope | language |
ontology | protocol | conversation-id)>

<!ELEMENT sender (agentname)>
<!ATTLIST sender link CDATA #REQUIRED >

<!ELEMENT receiver (agentname)>
<!ATTLIST receiver link CDATA #REQUIRED >

<!ELEMENT content (#PCDATA)>
<!ATTLIST content link CDATA #REQUIRED >

<!ELEMENT reply-with (#PCDATA)>
<!ELEMENT reply-by (#PCDATA)>
<!ELEMENT in-reply-to (#PCDATA)>
<!ATTLIST in-reply-to link CDATA #REQUIRED >

<!ELEMENT envelope (#PCDATA)>
<!ELEMENT language (#PCDATA)>
<!ATTLIST language link CDATA #REQUIRED >

<!ELEMENT ontology (#PCDATA)>
<!ATTLIST ontology link CDATA #REQUIRED >

<!ELEMENT protocol (#PCDATA)>
<!ATTLIST protocol link CDATA #REQUIRED >

<!ELEMENT conversation-id (#PCDATA)>

<!ELEMENT agentname (#PCDATA)>
```

Fig. 3. A DTD for ACML. The DTD is in draft form.

Let $\mathcal{C}_1$ be a simple example CLP ruleset that contains the single rule `giveDiscount(percent5 , ?Cust) <- shopper(?Cust) and loyalCustomer(?Cust).`
, shown here in ASCII encoding. This rule says to give a 5% discount to loyal customers. The CLP ruleset $\mathcal{C}_1$ can be expressed in BRML as follows:

```
<?xml version="1.0"?>
<!DOCTYPE brml SYSTEM "brml.dtd">
<clp>
  <erule rulelabel="emptyLabel">
    <head>
      <cliteral predicate="giveDiscount">
        <function name="percent5"/>
        <variable name="?Cust"/>
      </cliteral>
    </head>
    <body>
      <and>
        <fcliteral predicate="shopper">
          <variable name="?Cust"/>
        </fcliteral>
        <fcliteral predicate="loyalCustomer">
          <variable name="?Cust"/>
        </fcliteral>
      </and>
    </body>
  </erule>
</clp>
```

Fig. 4. An example of a single-rule CLP ruleset expressed in BRML.

```
<?xml version="1.0" encoding="US-ASCII"?>

<!ENTITY % bool "yes|no">

<!ELEMENT clp (erule*, mutex*)>
<!ATTLIST documentation link CDATA #IMPLIED>

<!ELEMENT erule (head, body?)>
<!ATTLIST erule rulelabel CDATA #IMPLIED>

<!ELEMENT mutex (cliteral, cliteral)>

<!ELEMENT head (cliteral | and)>

<!ELEMENT body (fcliteral | and | or)>

<!ELEMENT cliteral ( (function|variable|string)* )>
<!ATTLIST cliteral predicate CDATA #REQUIRED>
<!ATTLIST cliteral cneg (%bool;) #IMPLIED>

<!ELEMENT fcliteral ( (function|variable|string)* )>
<!ATTLIST fcliteral predicate CDATA #REQUIRED>
<!ATTLIST fcliteral cneg (%bool;) #IMPLIED>
<!ATTLIST fcliteral fneg (%bool;) #IMPLIED>

<!ELEMENT and ((cliteral|fcliteral|and|or),
               (cliteral|fcliteral|and|or)+)>

<!ELEMENT or ((fcliteral|and|or), (fcliteral|and|or)+)>

<!ELEMENT function ((function|variable|string)*)>
<!ATTLIST function name CDATA #REQUIRED>

<!ELEMENT variable EMPTY>
<!ATTLIST variable name CDATA #REQUIRED>

<!ELEMENT string EMPTY>
<!ATTLIST string value CDATA #REQUIRED>
```

Fig. 5. A DTD for BRML. The DTD is in draft form.