The OO jDrew Reference Implementation of RuleML

Ball, M.; Boley, Harold; Hirtle, D.; Mei, J.; Spencer, Bruce

National Research Council Canada
Conseil national de recherches Canada

Institute for Information Technology
Institut de technologie de l'information

NRC · CNRC

The OO jDrew Reference Implementation of RuleML *

Ball, M., Boley, H., Hirtle, D., Mei, J., and Spencer, B.
November 2005

* published in the RuleML 2005 Conference Proceedings. November 10-12, 2005. Galway, Ireland. Springer LNCS 3791. pp. 218-223. NRC 48284.

Canada

The OO jDREW Reference Implementation of RuleML

Marcel Ball[1], Harold Boley[2], David Hirtle[1,2], Jing Mei[1,2], Bruce Spencer[2]

[1] Faculty of Computer Science
University of New Brunswick
Fredericton, NB, E3B 5A3, Canada
{maball, david.hirtle, jingmei.may}@gmail.com
[2]Institute for Information Technology - e-Business
National Research Council of Canada
Fredericton, NB, E3B 9W4, Canada
{Harold.Boley, David.Hirtle, Jing.Mei, Brunce.Spencer}@nrc.gc.ca

Abstract. This paper presents the open source reference implementation of RuleML based on modular XML Schema definitions and bidirectional OO jDREW interpreters written in Java. For the family of RuleML sublanguages, schema modularization and RDF rules are discussed. The central bidirectional interpreters are introduced via jDREW principles, and explained w.r.t. OO jDREW slots, types, OIDs, and extensions.

1 Introduction

The syntax of RuleML derivation rules has been defined by XML Schema definitions. The model-theoretic semantics of several RuleML sublanguages (including Datalog, Hornlog, and Folog) is defined in the classical way; for sublanguages with negation-as-failure, well-founded models have been proposed. We have implemented the operational semantics of Derivation RuleML using XSLT translators and the bidirectional interpreter (the OO jDREW rule engine) described in this paper. This reference implementation is available open source via the RuleML and jDREW websites.

2 Modular Schemas for a Family of RuleML Sublanguages

The top-level of the current family of RuleML sublanguages shows the major distinction between Derivation Rules, including Hornlog above Datalog, and Action Rules, including Production Rules. We focus here on various expressive classes of Derivation Rules and their XML Schema Definitions (XSDs) as described in the Modularization document. The most recent (public) schema specification of RuleML is always available at http://www.ruleml.org/spec.

2.1 Schema Modularization

We employ modular XSDs, using a content model-based approach to take advantage of inheritance between schemas. Each expressive class syntactically distinguishable via an XSD (such as Datalog vs. Hornlog) can thus be addressed by the URI of its XSD. This permits receivers of a rulebase to validate if it conforms to the specified expressive class, before applying any class-specific tools (such as a Datalog vs. Hornlog interpreter). Moreover, a syntactic class is associated with a semantic class (such as Datalog vs. Hornlog with a function-free vs. function-containing Herbrand model). The relationships between these elements of the model are either aggregation, e.g. "Datalog is part of Hornlog", or generalization, e.g. "Bindatalog is a Datalog".

From an implementation perspective, elementary non-standalone modules contain only element and/or attribute definitions and are not intended to be used directly for validation. They may, however, be used to create new document types by others wishing to "borrow" certain elements of RuleML. The actual sublanguages, on the other hand, are schema drivers composed in whole or in part of these modules or derived entirely from other schema drivers.

2.2 RDF Rules as Anchored, Slotted Datalog with Blank Nodes

As an important sublanguage example, the definition of RDF Rules can be introduced in the following steps:

- Datalog is a language corresponding to relational databases (ground facts without complex domains or "constructors") augmented by views (possibly recursive rules).
- Slots permit non-positional arguments. RuleML's user-level metarole 'slot' takes a name-filler pair, accommodating RDF properties.
- Anchors provide object identity via webizing through URIs. Such "URI grounding" is available in RuleML via dual attributes 'wlab' and 'wref', corresponding to RDF's 'about' and 'resource'.
- Blank Nodes are local aliases for existing individuals without need for global names. In RuleML, the F-logic/Flora-2 Skolem-constant approach [1] is used to notate, generate, and refer to Blank Nodes.

Illustrating an RDF-like Business Rule 1:

```
<Implies>
  <body>
    <And>
      <Atom>
        <oid><Var>x</Var></oid>
        <Rel>product</Rel>
        <slot><Ind wref=":price"/><Var>y</Var></slot>
        <slot><Ind wref=":weight"/><Var>z</Var></slot>
      </Atom>
```

```
      <Atom>
        <Rel wref="swrlb:greaterThan"/><Var>y</Var><Data>200</Data>
      </Atom>
      <Atom>
        <Rel wref="swrlb:lessThan"/><Var>z</Var><Data>50</Data>
      </Atom>
    </And>
  </body>
  <head>
    <Atom>
      <oid><Var>x</Var></oid>
      <Rel>product</Rel>
      <slot><Ind wref=":shipping"/><Data>0</Data></slot>
    </Atom>
  </head>
</Implies>
```

3 Bidirectional Interpreters in Java

As part of the implementation of RuleML, a system of bidirectional interpreters, was created in Java. In particular, the OO jDREW reasoning engine [3] contains two modes: a Bottom-Up (forward chaining) version, and a goal driven top-down (backward chaining) version that works in a fashion similar to most Prolog systems. Demo applications (interfaced through Java Web Start) are available at http://www.jdrew.org/oojdrew/demo.html, and the source has been made available for download. A Roadmap for Open Source OO jDREW Development has recently been outlined (http://mail.jdrew.org/pipermail/jdrew-all/2005-June/000001.html). Principles, specifics, and extensions of the features available in OO jDREW are detailed below.

3.1 jDREW Principles

The jDREW toolbox approach [2] provides the flexibility to quickly cope with changes to the implementation of the evolving RuleML standard. There are utilities in jDREW for various tasks: reading files of RuleML statements into the internal clause data structure, storing and manipulating clauses, unification of clauses according to the positions of the selected literals, a basic resolution engine, clause to clause subsumption and clause to clause-list subsumption, choice point managers, priority queues for various reasoning tasks, and readable top-level procedures.

Much of the flow of control is oriented around iterators, objects that maintain the state of a partially completed computation. Thus you pay as you go when you want the engine to perform the next step. The advantages of this architecture are its consistency and efficiency. There is a common interface for many different reasoning tasks, and there are few additional data structures introduced for

storing intermediate results, other than those required by the abstract reasoning procedure. For instance, in the bottom-up system, solutions are generated one-at-a-time, so asking for the next solution may cause the following steps: An iterator will be asked to select the next clause in the so-called 'new results' list that matches eligibility requirements (like not being already subsumed).

3.2 OO jDREW Slots

During the creation of the internal structures, the OO jDREW terms representing atoms and complex terms are normalized, producing the following order for the parameters: oid (object identifier), positional parameters (in their original order), the positional rest term, slotted parameters (in the encounter order), and finally the slotted rest term. Since the ordering of slots within RuleML atoms and complex terms does not carry information, any order can be imposed. In OO jDREW, the slots are ordered based upon the sequence in which they are initially encountered to permit the incremental addition of slots without any reordering.

By using such a normalized form we are able to implement an efficient unification algorithm that has time complexity O(m + n) (where m and n are the numbers of parameters), instead of O(m * n). In our algorithm we scan the two lists of parameters – matching up roles (and positions in the case of positional parameters) – and unify those parameters. If a role is present in one term but not in the other then the unmatched role is added to a list of rest terms in case the other has an appropriate rest term (otherwise unification fails). Such a collection of rest terms is used to dynamically generate a Plex (RuleML's generalization of a list) to be assigned to the corresponding rest parameter.

3.3 OO jDREW Types

OO jDREW includes an order-sorted type system as a core component. This type system allows the user to specify a partial order for a set of classes in RDFS via their (multiple) superclasses, allowing for the reuse of lightweight taxonomies of the Semantic Web. Currently, the system only models the taxonomic relationships between the classes, and cannot model properties with their domain and range restrictions. For example, the current system can model that 'Car' is a 'Motor Vehicle', but cannot model that a car must have a make, model, year, etc.

By building an order-sorted type system into OO jDREW we are able to restrict the search space to only those clauses that have the appropriate types specified for their parameters, leading to a faster and more robust system than one where types are reduced to unary predicate calls in the body.

Extensions to the type system are being considered that would expand its modeling ability. In particular, the user could define a signature using RDFS properties to specify that certain slots must be present for a typing to be valid. We would then be able to prescribe that 'Car' has slots for at least make, model, year, which is not possible in the current system.

3.4 OO jDREW OIDs

The current implementation of OO jDREW, version 0.88, has a preliminary implementation of object identifiers (OIDs). Currently, OIDs are only supported in an atomic formula (<Atom> in RuleML), either as a fact or as part of a rule (<Implies> in RuleML). In this version only symbolic names are allowed as OIDs. The URI-valued wref and wlab attributes, which are part of the RuleML specification, are currently not supported; this is primarily due to W3C issues with the normalization of URIs, creating difficulties in determining what URIs should be considered to be equivalent.

The open source roadmap for OO jDREW includes plans to extend support for OIDs beyond their current level. It is envisioned that by the release of version 0.89, OIDs will be supported on levels other than atoms, such as for connectives and performatives. Additionally, wlab and wref should be supported with a preliminary URI normalization system, possibly implemented in OO RuleML [4] itself.

3.5 OO jDREW Extensions

Negation-as-failure (Naf) has first been implemented in OO jDREW TD, and recently introduced into OO jDREW BU for stratified programs. In bottom-up mode, Naf attempts to look up its argument atom via a unifying fact (when no other rule is applicable). If this look-up succeeds, hence the Naf fails, then this rule will be deleted from the given list, else the rule will be partially evaluated into one without Naf.

Equivalence classes have been implemented in OO jDREW BU for the sublanguage datalogeq (Datalog with Equality). For equality ground facts, a newly-built data structure called EqualTable is used to map all equal individuals to one equivalence class. For each equivalence class, we append a fresh symbol to the original OO jDREW SymbolTable, and all equal individuals are redirected to this new symbol. That is, the process of unification and resolution will deal with this new symbol, representing the equivalence class as a whole.

Illustrating Naf and Equal with a Datalog-like Business Rule 2:

```
<Implies>
  <head>
    <Atom>
      <Rel>discount</Rel>
      <Var>customer</Var><Var>product</Var><Ind>5.0 percent</Ind>
    </Atom>
  </head>
  <body>
    <And>
      <Atom><Rel>premium</Rel><Var>customer</Var></Atom>
      <Atom><Rel>onsale</Rel><Var>product</Var></Atom>
      <Naf>
```

```
        <Atom><Rel>special</Rel><Var>product</Var></Atom>
      </Naf>
    </And>
  </body>
</Implies>

<Equal><Ind>fatherOFtom</Ind><Ind>bob</Ind></Equal>
<Equal><Ind>fatherOFtom</Ind><Ind>uncleOFmary</Ind></Equal>
<Atom><Rel>premium</Rel><Ind>bob</Ind></Atom>
<Atom><Rel>onsale</Rel><Ind>clothes</Ind></Atom>

Results: discount("bob", clothes, "5.0 percent").
         discount("uncleOFmary", clothes, "5.0 percent").
         discount("fatherOFtom", clothes, "5.0 percent").
```

A detailed design of an indexing system has been completed for OO jDREW (http://www.jdrew.org/oojdrew/docs/OOjDREWIndexDesign.pdf) that will index the combined positional and slotted parameters on the top-level of RuleML atoms, along with their associated rest parameters. Once implemented, it should provide a significant increase in efficiency for the common cases, without creating too much overhead for the more unusual boundary cases.

4 Conclusions

RuleML has an open source implementation that is freely available and maintained as the standard evolves. The syntax of the family of sublanguages is specified by modular XML Schema definitions. The operational semantics of RuleML is implemented by a set of bidirectional interpreters (OO jDREW) written in Java for cross-platform compatibility. For interoperability with other standards, translators have also been realized, primarily via XSLT.

References

1. Reasoning about Anonymous Resources and Meta Statements on the Semantic Web, G. Yang and M. Kifer, In Journal on Data Semantics, Volume 1, Pages 69-97, 2003.
2. The Design of j-DREW: A Deductive Reasoning Engine for the Web, B. Spencer, In Proceedings of the First CologNET Workshop on Component-Based Software Development and Implementation Technology for Computational Logic Systems. CBD ITCLS 2002, Madrid, Spain. September 20, 2002. pp. 155-166.
3. OO jDREW: Design and Implementation of a Reasoning Engine for the Semantic Web, Marcel Ball, CS 4997, Faculty of Computer Science, University of New Brunswick, Fredericton, Canada, April 2005.
4. Object-Oriented RuleML: User-Level Roles, URI Grounded Clauses, and Order-Sorted Terms, H. Boley, In Proc. Rules and Rule Markup Languages for the Semantic Web (RuleML-2003). Sanibel Island, Florida, LNCS 2876, Springer-Verlag, October 2003.