

R2ML - The REVERSE I1 Rule Markup Language*

Gerd Wagner
Institute of Informatics
Brandenburg University of
Technology at Cottbus
03046 Walther Pauer Str.2,
Cottbus, Germany
G.Wagner@tu-cottbus.de

Adrian Giurca
Institute of Informatics
Brandenburg University of
Technology at Cottbus
03046 Walther Pauer Str.2,
Cottbus, Germany
Giurca@tu-cottbus.de

Sergey Lukichev
Institute of Informatics
Brandenburg University of
Technology at Cottbus
03046 Walther Pauer Str.2,
Cottbus, Germany
Lukichev@tu-cottbus.de

ABSTRACT

This study concerning R2ML (REVERSE I1 Rule Markup Language) an interchange format for rules integrating the *Rule Markup Language* (RuleML) with the *Semantic Web Rule Language* (SWRL) as well as the *Object Constraint Language* (OCL). These languages provide a *rich syntax* for expressing rules. This means that they support conceptual distinctions, such as distinguishing different types of terms and atoms, which are not present in standard predicate logic. The interchange format is *usable* in the sense that it allows structure-preserving markup of all constructs of these different languages and does not force users to translate their rule expressions to a completely different language paradigm such as having to transform a function into a functional predicate.

R2ML is serialization language for the vizual tool Strelka built in top of Fujaba environment.

Keywords: Rules, rule markup languages, integrity rules derivation rules, production rules, rule metamodels, OCL, RuleML, SWRL.

In R2ML, at that moment, we consider three kinds of rules in this report: integrity rules, derivation rules and production rules. We define rule concepts with the help of MOF/UML, a subset of the UML class modeling language proposed by the Object Management Group (OMG).

1. BASIC CONTENT VOCABULARY

The user-defined content vocabulary includes:

- user-defined **object names**;

*This research has been funded by the European Commission and by the Swiss State Secretariat for Education and Research within the 6th Framework Programme project REVERSE no.506779

- user-defined **object function names** comprising role function names and object property names;
- user-defined **data function names** comprising attributes and data operations;
- user-defined **noun concept names** standing for general noun concepts, among which we distinguish *object types* ('classes') and '*datatypes*';
- user-defined **verb concept names**, called 'predicate symbols' in traditional logic, standing for general verb concepts, or *predicates*, among which we distinguish *properties* and *associations*; properties are either *attributes*, if they are data-valued, or *reference properties*, if they are object-valued.

In Web languages such as RDF and OWL, all these names are globally unique standard identifiers in the form of URI references. One of the goals of R2ML is to comply with important Semantic Web standards like RDF(S) and OWL. In particular, R2ML accommodates the data type concept of RDF.

User-defined noun concepts comprise **classes** (or *object types*). Usually, any object or object variable belongs to a class.

A *class* in R2ML is an URI reference. A class is a type entity for R2ML objects and object variables.

The datatype language consists of a set of predefined datatype names, including the name `rdfs:Literal` standing for the generic built-in datatype of all Unicode strings. Each predefined datatype name is associated with

- a set of **data literals**, which are Unicode strings;
- a set of **datatype function names**;
- a set of **datatype predicate names**.

A *datatype* in R2ML is an `rdfs:Literal` or an user defined datatype (subclass of `rdf:TypedLiteral`) referenced by an URI reference. A datatype is a type entity for R2ML data values and data variables.

User-defined verb concepts comprise **properties** and **associations**. Properties are either *attributes*, if they are data-valued, or *reference properties*, if they are object-valued.

Notice that we use an attribute name both as the name of a function and as the name of the corresponding functional predicate. Likewise, we use a reference property name both as the name of a property predicate and as the name of the corresponding role function. This kind of naming liberty, which is supported by RDF and Common Logic, helps to switch between functional and relational languages.

A *reference property* in R2ML is an URI reference. A reference property corresponds to the `rdf:property` property from RDF. It is used in `R2ML ReferencePropertyAtom`.

A *datatype predicate* in R2ML is an URI reference. A datatype predicate accommodates the SWRL concept of a built-in predicate (predicates that from `http://www.w3.org/2003/11/swrlb` namespace).

In R2ML individual terms are either *object terms* standing for *objects*, or *data terms* standing for *data values*. The concrete syntax of first-order non-Boolean OCL expressions can be directly mapped to our abstract concepts of *ObjectTerm* and *DataTerm* which can be viewed as a predicate-logic-based reconstruction of the standard OCL abstract syntax.

As in RuleML [2], R2ML provides the concept of a *variable* but as in almost all programming languages, distinguishes between *object* and *data* variables i.e. between references that are instantiated with objects or with data values.

As in RuleML, R2ML defines the concept of an individual (constant) but distinguishes between objects and data. Following the terminology from UML we define the concepts of an *object name* and *data value*. An *ObjectName* contains an optional reference to a class which is its type. R2ML allows both typed and un-typed individuals. As in RDF a *DataValue* consists in a lexical value and a type which is an RDF datatype or a user-defined datatype (subclass of `rdfs:Literal`).

In order to support common fact types of natural language directly, it is important to have *n*-ary predicates (for $n > 2$).

An *association predicate* in R2ML is an URI reference. An association predicate corresponds to the standard Datalog predicates. It is used in `AssociationPredicateAtom`.

2. FUNCTIONAL CONTENT VOCABULARY

In R2ML a *datatype function* is a functional symbol identified by a URI reference.

A *data operation* is a special type of user-defined function that corresponds to a data-valued operation in a UML class model.

An *attribute* is a special type of user-defined function that corresponds to a data-valued property in a UML class model. In Figure ??, `reservationDate` is an attribute of the class `Rental`.

3. FUNCTIONAL TERMS

A data term is either a data variable, a data value, or a *data function term*, which can be of three different types:

1. A *datatype function term* formed with the help of a datatype function that comes with the corresponding datatype.
2. An *attribute function term* formed with the help of a user-defined attribute.
3. A *data operation term* formed with the help of a user-defined data operation.

User-defined data functions are either *attributes* or *data operations*. They are used in *data terms*. R2ML defines *AttributeFunctionTerm*, *DataOperationTerm* and *DatatypeFunctionTerm*.

User-defined object functions are either *role functions* or *object operations*. They are used in *object terms*.

An **object operation** is a special type of user-defined function that corresponds to an object-valued operation in a UML class model. For example, in Figure ??, the operation `getLastRental()` defines an object operation.

4. ATOMS

The basic constituent of a rule is the *atom*. In R2ML we define metamodelling for atoms, which are compatible with all important concepts of OWL, SWRL and RuleML.

4.1 Basic Atoms

An *object classification atom* refers to a class and consists of an object term.

A *data classification atom* consists of a data term and refers to a data type.

As in SWRL [3], R2ML supports the concepts of equality and inequality atoms.

An *equality atom* or *inequality atom*, is composed of two or more object terms.

A *built-in data predicate atom* refers to a datatype predicate, and consists of a number of data terms.

4.2 Relational Atoms

An *attribution atom* consists of an object term as "subject", and a data term as "value".

A *reference property atom* associates object terms as "subjects" with other object terms as "objects".

An *association atom* is constructed using an *n*-ary predicate as association predicate, a collection of data terms as "data arguments" and a collection of object terms as "object arguments".

Following RDF [?] and OWL [?], we adopt the concept of an *object description atom*.

An *object description atom* refers to a class as a base type and to zero or more classes as categories, and consists of a number of property/term pairs (attribute data term pairs and reference property object term pairs). Any instance of such atom refers to one particular object, that is referenced by an objectID, if it is not anonymous.

5. FORMULAS

R2ML provides two abstract concepts for formulas: the concept of *AndOrNafNegFormula*, which represents the most general quantifier free logical formula with weak and strong negations, and the concept of *LogicalFormula*, which corresponds to a general quantified first order formula.

6. ACTIONS

The REVERSE Rule Markup Language offer support both for production rules and reaction rules. With this respect it define the concept of an *action*. Following the OMG Production Rule Representation submission, an action is either an *InvokeActionExpression* or an *AssignActionExpression* or a *CreateActionExpression* or a *DeleteActionExpression*.

InvokeAction refers to an UML *Operation* and contains a list of arguments. This action invokes an operation with a list of arguments.

AssignAction refers to an UML *Property* and contains a *DataTerm*. This action assigns a value to a property.

CreateAction refers to an UML *Class* and to a list of *ReferencePropertyObjectTermPair* and/or *AttributeDataTermPair*.

DeleteAction refers to an UML *Class* and contains an *ObjectVariable*. This action removes an instance of the *Class*.

7. RULES

7.1 Integrity Rules

An integrity rule consist in a constraint that is a general logical formula without free variables.

EXAMPLE 1. *We use examples from the EU-Rent case study*¹.

If a rental is not an one way rental then return branch of rental must be the same as pick-up branch of rental

```
<rew:AlethicIntegrityRule rew:id="IT1001">
  <rew:Constraint>
    <rew:Implication>
      <rew:Conjunction>
        <rew:ObjectClassificationAtom
          rew:classID="srv:OneWayRental"
          rew:isNegated="true">
          <rew:ObjectVariable rew:name="rental"
            rew:classID="srv:Rental"/>
        </rew:ObjectClassificationAtom>
      </rew:Conjunction>
    </rew:Implication>
  </rew:Constraint>
</rew:AlethicIntegrityRule>
```

¹EU-Rent case study at the European Business Rules Conference web site <http://www.eurobizrules.org/eurentcs/eurent.htm>

```
<rew:EqualityAtom>
  <rew:RoleFunctionTerm
    rew:refPropertyID="srv:returnBranch">
    <rew:argument>
      <rew:ObjectVariable rew:name="rental"
        rew:classID="srv:Rental"/>
    </rew:argument>
  </rew:RoleFunctionTerm>
  <rew:RoleFunctionTerm
    rew:refPropertyID="srv:pickupBranch">
    <rew:argument>
      <rew:ObjectVariable rew:name="rental"
        rew:classID="srv:Rental"/>
    </rew:argument>
  </rew:RoleFunctionTerm>
</rew:EqualityAtom>
</rew:Conjunction>
</rew:Implication>
</rew:Constraint>
</rew:AlethicIntegrityRule>
```

7.2 Derivation Rules

Derivation Rules, have "conditions" and "conclusions". In R2ML framework the conditions of a derivation rule are *AndOrNafNegFormula*. Conclusions are restricted to *AndOrNegFormula* without NAF.

EXAMPLE 2. *If a rental car has then last maintenance date older than 90 days or the service reading greater than 5000km then it is a rental car scheduled for service.*

```
<rew:DerivationRule rew:id="dr1111"
  xmlns:srv="http://www.services.org/EU-Rent/">
  <rew:conditions>
    <rew:Disjunction>
      <rew>DataPredicateAtom rew:dataPredicateID="ge">
        <rew:AttributeFunctionTerm
          rew:attributeID="srv:last_maintenance_date">
          <rew:argument>
            <rew:ObjectVariable rew:name="rentalCar"
              rew:classID="srv:RentalCar"/>
          </rew:argument>
        </rew:AttributeFunctionTerm>
      <rew:TypedLiteral rew:typeLiteral="xs:positiveInteger"
        rew:lexicalValue="90"/>
    </rew>DataPredicateAtom>
      <rew>DataPredicateAtom rew:dataPredicateID="ge">
        <rew:AttributeFunctionTerm rew:attributeID="srv:service_reading">
          <rew:argument>
            <rew:ObjectVariable rew:name="rentalCar"
              rew:classID="srv:RentalCar"/>
          </rew:argument>
        </rew:AttributeFunctionTerm>
      <rew:TypedLiteral rew:typeLiteral="xs:positiveInteger"
        rew:lexicalValue="5000"/>
    </rew>DataPredicateAtom>
    </rew:Disjunction>
  </rew:conditions>
  <rew:conclusion>
    <rew:ObjectClassificationAtom
      rew:classID="srv:RentalCarScheduledForService">
      <rew:ObjectVariable rew:name="rentalCar"/>
    </rew:ObjectClassificationAtom>
  </rew:conclusion>
</rew:DerivationRule>
```

7.3 Production Rules

Production Rules, have "conditions" and "post-conditions". The conditions and post-conditions of a production rule are *LogicalFormula*. A production rule may execute an *Action*.

EXAMPLE 3. *Assign Discount*

R2ML	SWRL
object variable	individual variable
object name	individual (owl:Individual)
role function term n/a	
data variable	data variable
data value (rdfs:Literal)	data literal (owlx:datatype)
data operation term	n/a
attribute function term	n/a
datatype function term	n/a
data operation term	n/a

Table 1: Comparison by supported terms

If the customer order has the value greater than 1000 and the customer is not a "gold" customer, then assign a discount of 10%.

```

<rew:ProductionRule xmlns:srv="http://www.services.org/EU-Rent/">
  <rew:conditions>
    <rew:EqualityAtom>
      <rew:ObjectVariable rew:name="y"
        rew:classID="srv:Customer"/>
      <rew:RoleFunctionTerm rew:refPropertyID="srv:customer">
        <rew:argument>
          <rew:ObjectVariable rew:name="x"
            rew:classID="srv:Order"/>
        </rew:argument>
      </rew:RoleFunctionTerm>
    </rew:EqualityAtom>
    <rew:InequalityAtom>
      <rew:RoleFunctionTerm rew:refPropertyID="srv:type">
        <rew:argument>
          <rew:ObjectVariable rew:name="y"
            rew:classID="srv:Customer"/>
        </rew:argument>
      </rew:RoleFunctionTerm>
      <rew:ObjectName rew:objectID="srv:gold"/>
    </rew:InequalityAtom>
    <rew>DataPredicateAtom rew:dataPredicateID="srv:ge">
      <rew:AttributeFunctionTerm rew:attributeID="srv:value">
        <rew:argument>
          <rew:ObjectVariable rew:name="x" rew:classID="srv:Order"/>
        </rew:argument>
      </rew:AttributeFunctionTerm>
      <rew:TypedLiteral rew:typeLiteral="xs:positiveInteger"
        rew:lexicalValue="1000"/>
    </rew>DataPredicateAtom>
  </rew:conditions>
  <rew:producedAction>
    <rew:AssignAction rew:propertyID="srv:discount">
      <rew:contextArgument>
        <rew:ObjectVariable rew:name="x" rew:classID="srv:Order"/>
      </rew:contextArgument>
      <rew:TypedLiteral rew:lexicalValue="10"
        rew:typeLiteral="xs:positiveInteger"/>
    </rew:AssignAction>
  </rew:producedAction>
</rew:ProductionRule>

```

8. COMPARISON WITH SWRL

9. FUTURE WORK

Next steps in the development of R2ML concern:

- Finalization of the schema.
- XSLT and CSS for rule publishing on the web.
- Integrating rules in server side programming.
- Complete integration with URML and Strelka (as a serialization language).

R2ML	SWRL
ObjectClassificationAtom	classAtom
AttributionAtom	datavaluedPropertyAtom
ReferencePropertyAtom	individualPropertyAtom
DataClassificationAtom	datarangeAtom
EqualityAtom	sameIndividualAtom
InequalityAtom	differentIndividualsAtom
DataPredicateAtom	builtinAtom
AssociationAtom	n/a
ObjectDescriptionAtom	n/a

Table 2: Comparison by supported atoms

- Extending R2ML for supporting other rule languages developed in REVERSE.

10. REFERENCES

- [1] Wagner, G., How to Design a General Rule Markup Language, Proceedings of the Workshop XML Technologies for the Semantic Web (XSW 2002), HU Berlin, Insti tut fur Informatik, June 2002, Lecture Notes in Informatics, Gesellschaft f. Informatik.
- [2] Wagner, G., Antoniou, G., Tabet, S., and Boley, H., The Abstract Syntax of RuleML - Towards a General Web Rule Language Framework, Rule Markup Initiative (RuleML), <http://www.ruleml.org>
- [3] Horrocks I., Patel-Schneider P. F., Bell Labs Research, Boley H., Tabet S., Groszof B., Dean M., SWRL: A Semantic Web Rule Language Combining OWL and RuleML, W3C Member Submission 21 May 2004, <http://www.w3.org/Submission/SWRL/>
- [4] W3C Workgroup on RIF Charter, <http://www.w3.org/2005/rules/wg/charter>
- [5] G. Wagner, A. Giurca, S. Lukichev (2005). R2ML: A General Approach for Marking up Rules, Dagstuhl Seminar Proceedings 05371, in F. Bry, F. Fages, M. Marchiori, H. Ohlbach (Eds.) Principles and Practices of Semantic Web Reasoning, <http://drops.dagstuhl.de/opus/volltexte/2006/479/>
- [6] EU-Rent case study at the European Business Rules Conference web site <http://www.eurobizrules.org/eurentcs/eurent.htm>