RuleML Intro Examples and More Syntax Details

Harold Boley*, NRC IIT e-Business
http://www.dfki.de/~boley

Benjamin Grosof, MIT Sloan
http://www.mit.edu/~bgrosof

(with help from Bruce Spencer, Steve Ross-Talbot, Said Tabet, and Gerd Wagner)

* On leave from DFKI GmbH

Presentation for
Joint US/EU ad hoc Agent Markup Language Committee
Meeting of 29 October 2002
"The **discount** for a *customer* buying a *product* is **5.0 percent** if the *customer* is **premium** and the *product* is **regular**.

discount(\(?customer, ?product, "5.0 percent"") ← premium(?customer) ∧ regular(?product);

```xml
<imp>
  <_head>
    <atom>
      <_opr><rel>discount</rel></_opr>
      <tup><var>customer</var></tup>
      <var>product</var></tup>
      <ind>5.0 percent</ind>
    </atom>
  </_head>
  <_body>
    <and>
      <atom>
        <_opr><rel>premium</rel></_opr>
        <tup><var>customer</var></tup>
      </atom>
      <atom>
        <_opr><rel>regular</rel></_opr>
        <tup><var>product</var></tup>
      </atom>
    </and>
  </_body>
</imp>
```

**tup** is an ordered tuple.
Non-Positional RuleML Via the Name-Giving Metarole _r (I)

The minimal 'metarole' (_r) representation of the non-positional, RDF-like Jess fact

(automobile (make Ford) (model Explorer) (year 1999))

in RuleML 0.82 with user roles named (n) by XML attributes:

```xml
<fact>
  <_head>
    <atom>
      <_opr><rel>automobile</rel></_opr>
      <_r n="make"><ind>Ford</ind></_r>
      <_r n="model"><ind>Explorer</ind></_r>
      <_r n="year"><ind>1999</ind></_r>
    </atom>
  </_head>
</fact>
```
Non-Positional RuleML Via the Name-Giving Metarole _r (II)

This 'non-positional RuleML' notation corresponds to the 'positionalized' ruleml-datalog notation

```xml
<fact>
  <_head>
    <atom>
      <_opr><rel>automobile</rel></_opr>
      <ind>Ford</ind>
      <ind>Explorer</ind>
      <ind>1999</ind>
    </atom>
  </_head>
</fact>
```

if the 'roles' of the **make**, **model**, and **year** positions are remembered somewhere else (signature declaration)
From Limited Natural Language to Horn Logic

English-subset Business Rules:

"The discount for a customer buying a product is 5.0 percent if the customer is premium and the product is regular."

"The discount for a customer buying a product is 7.5 percent if the customer is premium and the product is luxury."

... 

Prolog-like formalization (syntax generated from XML):

```prolog
discount(_customer, _product, 5.0 percent) :- premium(_customer), regular(_product).
discount(_customer, _product, 7.5 percent) :- premium(_customer), luxury(_product).
premium(_customer) :- spending(_customer, min 5000 euro, previous year).
luxury(Porsche).
regular(Honda).
spending(Peter Miller, min 5000 euro, previous year).
```
Structure of the RuleML DTD Hierarchy

- Our system of DTDs (current version: 0.8) uses a modularization approach similar to XHTML in order to accommodate the various rule subcommunities.
- The evolving hierarchy of RuleML DTDs forms a partial order with ruleml as the greatest element (a ruleml-rooted DAG) -- many ‘smallest’ elements.
- Each DTD node in the hierarchy (conformance “lattice”) corresponds to a specific RuleML sublanguage, syntactically and semantically:
  - ‘Union’ (join) of sublanguages reached via outgoing links: to smaller or equal nodes below.
  - ‘Intersection’ (meet) of sublanguages via incoming links: from greater or equal nodes above.
Each DTD/XSD is fairly short – a few pages -- e.g., Situated Courteous
(from SweetRules v1)
Situated Courteous DTD, cont’d

<!-- syntax for courteous and situated follows --->

<!ELEMENT mutex ((_oppo, _mgiv?) | (_mgiv, _oppo))>
<!ELEMENT _oppo (ando)>
<!ELEMENT _mgiv (fclit | andb | orb)>
<!ELEMENT ando (clit, clit)>

<!ENTITY % bind "bound|free">
<!ELEMENT sens ((_opr, ((_aproc, _modli?) | (_modli, _aproc))) | (_aproc, ((_opr, _modli?) | (_modli, _opr))) | (_modli, ((_aproc, _opr) | (_opr, _aproc))))>
<!ELEMENT effe ((_opr, _aproc) | (_aproc, _opr))>
<!ELEMENT _aproc (jproc | uproc)>
<!ELEMENT uproc (#PCDATA)>
<!ATTLIST uproc href %URI; #IMPLIED>
<!ELEMENT jproc ((clas, ((meth, path?) | (path, meth))) | (meth, ((clas, path?) | (path, clas))) | (path, ((meth, clas) | (clas, meth))))>
<!ELEMENT path (#PCDATA)>
<!ATTLIST path href %URI; #IMPLIED>
<!ELEMENT clas (#PCDATA)>
<!ATTLIST clas href %URI; #IMPLIED>
<!ELEMENT meth (#PCDATA)>
<!ATTLIST meth href %URI; #IMPLIED>
<!ELEMENT _modli ((bmode)*)>
<!ELEMENT bmode EMPTY>
<!ATTLIST bmode val (%bind;) "free"
More: Pointers

- RuleML DTD 0.8, a system of DTDs, is available at http://www.dfki.de/ruleml/indtd0.8.html; sample rulebases at http://www.dfki.de/ruleml/0.8/exa, use cases at http://www.dfki.de/ruleml/library