

Markup F-Logic with R2ML*

Marco Pehla, (marco.pehla@googlemail.com)

August 7, 2006

Contents

1	Introduction	1
2	How could we markup F-Logic with R2ML ?	1
2.1	Markup of facts	1
2.2	Markup of rules	2
2.3	Markup of queries	3
3	Built-ins	4
3.1	Comparison	4
3.2	Math	4
3.3	String	5
3.3.1	Concatenation	5
4	Appendix	6
4.1	built-ins table	6

1 Introduction

This paper should give a little overview how F-Logic could be marked up in R2ML . Therefore we use some of the F-Logic examples from the paper *How to Write F-Logic Programs*¹.

2 How could we markup F-Logic with R2ML ?

2.1 Markup of facts

An expression like `abraham:man` could be easily marked up as an `ObjectClassificationAtom` in R2ML. The corresponding source code is:

```
<r2ml:ObjectClassificationAtom r2ml:class="Man">
  <r2ml:ObjectName r2ml:object="abraham"/>
</r2ml:ObjectClassificationAtom>
```

*<http://www.rewerse.net/I1/>

¹http://www.ontoprise.de/documents/tutorial_flogic.pdf

Longer F-Logic expressions could be separated into two or more simpler expressions. The following simple example explain you how.

```
ishmael:man[father->abraham; mother->hagar:woman].
```

```
ishmael:man[father->abraham].
ishmael:man[mother->hagar:woman].
```

If we first separate such expressions then we could markup these afterwards in R2ML as the following. :

```
<r2ml:ReferencePropertyAtom r2ml:referenceProperty="father">
  <r2ml:subject>
    <r2ml:ObjectName r2ml:object="ishmael" r2ml:class="Man"/>
  </r2ml:subject>
  <r2ml:object>
    <r2ml:ObjectName r2ml:object="abraham"/>
  </r2ml:object>
</r2ml:ReferencePropertyAtom>

<r2ml:ReferencePropertyAtom r2ml:referenceProperty="mother">
  <r2ml:subject>
    <r2ml:ObjectName r2ml:object="ishmael" r2ml:class="Man"/>
  </r2ml:subject>
  <r2ml:object>
    <r2ml:ObjectName r2ml:object="hagar" r2ml:class="Woman"/>
  </r2ml:object>
</r2ml:ReferencePropertyAtom>
```

2.2 Markup of rules

Let's take a simple rule.

```
FORALL X,Y X[son->>Y] <- Y:man[father->X].
```

If a man Y has a father X then X has a son Y.

To markup these we use a `ObjectVariable` for X, Y, the single-valued method call `Y:man[father->X]` and the multi-valued method call `X[son->>Y]`.

```
...
<r2ml:conditions>
...
  <r2ml:ReferencePropertyAtom r2ml:referenceProperty="father">
    <r2ml:subject>
      <r2ml:ObjectVariable r2ml:name="Y" r2ml:class="Man"/>
    </r2ml:subject>
    <r2ml:object>
      <r2ml:ObjectVariable r2ml:name="X"/>
    </r2ml:object>
  </r2ml:ReferencePropertyAtom>
</r2ml:conditions>
```

```

<r2ml:conclusion>
  <r2ml:ReferencePropertyAtom r2ml:referenceProperty="son">
    <r2ml:subject>
      <r2ml:ObjectVariable r2ml:name="X"/>
    </r2ml:subject>
  </r2ml:object>
  <r2ml:ObjectVariable r2ml:name="Y" r2ml:isMultivalued="true"/>
</r2ml:ReferencePropertyAtom>
</r2ml:conclusion>

```

As you could see above, the head of the F-Logic rule is marked up inside the `<r2ml:conditions>` tags. The body of the rule is marked up between the `<r2ml:conclusion>` tags. For the multi-valued method call `X[son->>Y]` we use the `r2ml:isMultivalued` attribute with the value `true` inside the `ObjectVariable`. For the single-valued method call these attribute is mandatory. If we don't use it then the default value is `false`.

2.3 Markup of queries

Queries are a special kind of rules just with a body but without any head. For e.g. the following.

```
FORALL Y <- jacob[ancestor->>Y:woman].
```

All ancestors Y of jacob that were been woman's.

```

<r2ml:conditions>
  <r2ml:ObjectClassificationAtom r2ml:class="Woman">
    <r2ml:ObjectVariable r2ml:name="Y"/>
  </r2ml:ObjectClassificationAtom>
</r2ml:conditions>
<r2ml:conclusion>
  <r2ml:ReferencePropertyAtom r2ml:referenceProperty="ancestor">
    <r2ml:subject>
      <r2ml:ObjectName r2ml:object="jacob"/>
    </r2ml:subject>
    <r2ml:object>
      <r2ml:ObjectVariable r2ml:name="Y" r2ml:class="Woman"
        r2ml:isMultivalued="true"/>
    </r2ml:object>
  </r2ml:ReferencePropertyAtom>
</r2ml:conclusion>

```

In F-Logic there is no head in a query, but in R2ML we need at least one condition. We can help us with a double declaration to markup this special kind of rule. In this example we marked up the `r2ml:ObjectVariable` twice.

3 Built-ins

3.1 Comparison

F-Logic has some built-in features for comparison. You could use SWRL within R2ML to markup these. Let's have a look at the following example.

```
FORALL X,Y,Z <- jacob[son@(X,Y)->>Z] AND less(Y,4).
```

(Example 6.1)

The built-in `less(Y,4)` could be marked up in R2ML with the help of SWRL. Therefore we have to define at first the SWRL built-in namespace² inside of our rule element.

```
<r2ml:DerivationRule r2ml:id="DR006"
                    xmlns:swrlb="http://www.w3.org/2003/11/swrlb">
```

Now we are able to markup the expression `less(Y,4)` with the help of the SWRL built-ins.

```
<r2ml:DatatypePredicateAtom r2ml:datatypePredicate="swrlb:lessThan">
  <r2ml:dataArguments>
    <r2ml:DataVariable r2ml:name="y"/>
    <r2ml:TypedLiteral r2ml:lexicalValue="4"
                      r2ml:datatype="xs:positiveInteger"/>
  </r2ml:dataArguments>
</r2ml:DatatypePredicateAtom>
```

As you can see we are using the `swrlb:lessThan` as value in the `r2ml:datatypePredicate` attribute.

3.2 Math

Mathematical expressions like `B is A+1` could be marked up with the help of SWRL like the following.

```
<r2ml:DatatypeFunctionTerm r2ml:datatypeFunction="swrlb:add">
  <r2ml:dataArguments>
    <r2ml:DataVariable r2ml:name="B"/>
    <r2ml:DataVariable r2ml:name="A"/>
    <r2ml:TypedLiteral r2ml:lexicalValue="1"
                      r2ml:datatype="xs:positiveInteger"/>
  </r2ml:dataArguments>
</r2ml:DatatypeFunctionTerm>
```

The first element inside of the `r2ml:dataArguments` tag has to be the result of the operation declared in the `r2ml:datatypeFunction` attribute to verify the expression to **true**. The markup for `swrlb:subtract`, `swrlb:multiply` and `swrlb:divide` have a similar markup.

²<http://www.w3.org/2003/11/swrlb>

3.3 String

3.3.1 Concatenation

The F-Logic expression `concat("a","b","ab")` evaluates to **true** because the string `ab` is a concatenation of the strings `a` and `b`.

```
<r2ml:DatatypePredicateAtom r2ml:datatypePredicate="swrlb:stringConcat">
  <r2ml:dataArguments>
    <r2ml:TypedLiteral r2ml:lexicalValue="ab" r2ml:datatype="xs:string"/>
    <r2ml:TypedLiteral r2ml:lexicalValue="a" r2ml:datatype="xs:string"/>
    <r2ml:TypedLiteral r2ml:lexicalValue="b" r2ml:datatype="xs:string"/>
  </r2ml:dataArguments>
</r2ml:DatatypePredicateAtom>
```

4 Appendix

4.1 built-ins table

F-Logic built-in	SWRL built-in	description
less	swrlb:lessThan	no.1 is less than no.2
lessorequal	swrlb:lessThanOrEqual	no.1 is less than or equal no.2
greater	swrlb:greaterThan	no.1 is greater than no.2
greaterorequal	swrlb:greaterThanOrEqual	no.1 is greater than or equal no.2

Table 1: SWRL built-ins

For an complete description of all SWRL built-ins, please have a look at <http://www.w3.org/Submission/SWRL/#8.1> . Because at the moment SWRL is an W3C submission.