The Relational-Functional Markup Language RFML - Extended Abstract

Harold Boley, DFKI GmbH (boley@dfki.de)

"Knowledge on the Web" has become an increasingly important topic in computer science, crossing the boundaries of several traditional subfields such as information systems, artificial intelligence, programming languages, and computer networks. The topic is also impacting computer applications such as knowledge management, content distribution, and electronic business. Much of the knowledge on the Web constitutes definitions of relations and functions.

This talk gives a preliminary specification of the Relational-Functional Markup Language RFML, based on Relfun (http://www.relfun.org/) and XML (http://www.w3.org/XML/). The kernel of the Relational-Functional language is well-suited for XML markup of Web knowledge since it constitutes a uniform, rather small language, but - by virtue of the relational-functional integration - has sufficient expressive power for practical use. RFML can be seen as an XML application for integrated relational-functional information. This information consists of both relational (hn) and functional (ft) clauses, together defining a unified notion of operators, which can call each other in general ways. Because of the small kernel, it was also possible to keep the RFML document type definition small and open to various extensions.

The talk will argue for possible cross-fertilizations between the XML (schema and query) and declarative-programming communities. For instance, instead of separating XML-element assertion and querying, RFML incorporates both of these aspects in a uniform manner: asserted clauses often contain queries, and queries are also usable on an interactive top-level.

For example, the unconditional (ground) equation

```
pay(john,fred,17.95) = cheque
```

defining a 'pay' function, in RFML becomes the following markup:

```
<ft>
   <pattop>
      <con> pay </con>
      <con> john </con>
      <con> fred </con>
      <con> 17.95 </con>
   </pattop>
  <con>
     cheque
  </con>
</ft>
```

It can be queried (non-ground) directly via

```
<callop>
   <con> pay </con>
   <var> customer </var>
   <var> merchant </var>
   <var> price </var>
</callop>
```

binding the three variables to the corresponding constants in the definition pattern and returning the constant 'cheque'. It can also be queried indirectly by the Relfun conditional (non-ground) equation

```
acquire(Customer,Merchant,Product,Price) :-
   satisfied(Customer,Product,Price) &
   pay(Customer,Merchant,Price).
```

defining an 'acquire' operation, whose RFML version uniformly marks up the relational ('satisfied') and functional ('pay') subqueries via 'callop' subelements:

```
<ft>
   <pattop>
      <con> acquire </con>
      <var> customer </var>
      <var> merchant </var>
      <var> product </var>
      <var> price </var>
   </pattop>
   <callop>
      <con> satisfied </con>
      <var> customer </var>
      <var> product </var>
      <var> price </var>
   </callop>
   <callop>
      <con> pay </con>
      <var> customer </var>
      <var> merchant </var>
      <var> price </var>
   </callop>
</ft>
```

RFML has been implemented as a (Web-)output syntax for relational-functional knowledge bases and computations. Further descriptions and download information will be available http://www.relfun.org/rfml/.