This example demonstrates the use of data types and built ins. Note the use of *External* in applying built-in predicates. Also the use of *Group* to put together a number of rules.

The syntax of RIF is rather straightforward, though quite verbose (of course, there is also an XML-based syntax to support interchange between rule systems). Variable names begin with a question mark. And the symbols =, #, and ## are used to express equality, class membership, and subclass relationship, respectively.

The use of *frames* has a long tradition in object-oriented languages and knowledge representation, and has also been prominent in the area of rule languages (e.g., F-Logic). The basic idea is to represent objects as frames and their properties as *slots*. For example, we might have a class professor with slots such as name, office, phone, department, etc. Such information is expressed in RIF-BLD using the notation

```
oid[slot1 -> value1 ... slotn -> valuen]
```

### 5.6.3 Compatibility with RDF and OWL

A major feature of RIF is that it is compatible with the RDF and OWL standards. That is, one can reason with a combination of RIF, RDF, and OWL documents. Thus RIF facilitates the interchange of not just rules, but also RDF graphs and/or OWL axioms.

The basic idea of combining RIF with RDF is to represent RDF triples using RIF frame formulas; a triple *s p o* is represented as s[p -> o]. The semantic definitions are such that the triple is satisfied iff the corresponding RIF frame formula is, too. For example, if the RDF triple

```
ex:GoneWithTheWind ex:FilmYear ex:1939
```

is true, then so is the RIF fact

```
ex:GoneWithTheWind[ex:FilmYear -> ex:1939]
```