that the object denoted by the variable *?c* has the value denoted by the variable *?a* for the property *Chicken/age* that is defined in the example namespace *http://example.com/2008/joe#*.

```
<Frame>
   <object> <Var> c </Var> </object>
   <slot rif:ordered="yes">
      <Const type="rif:iri">
        http://example.com/2008/joe#Chicken/age
      </Const>
      <Var> a </Var>
   </slot>
</Frame>
```

Editor's Note: The example uses an XPath style for the key. How externally specified data models and their elements should be referenced is still under discussion (see ISSUE-37 (http://www.w3.org/2005/rules/wg/track/issues/37)).

5.2.2.6 External

In RIF-PRD, the `External` element is also used to serialize an externally defined atomic formula.

When it is a `ATOMIC` (as opposed to a `TERM`; that is, in particular, when it appears in a place where an `ATOMIC` is expected, and not a `TERM`), the `External` element contains one `content` element that contains one `Atom` element. The `Atom` element serializes the externally defined atom properly said:

```
<External>
   <content>
      Atom
   </content>
</External>
```

The `op Const` in the `Atom` element must be a symbol of type `rif:iri` that must uniquely identify the externally defined predicate to be applied to the `args TERMs`. It can be one of the builtin predicates specified for RIF dialects, as listed in section List of RIF Builtin Predicates and Functions of the RIF data types and builtins document, or it can be application specific. In the latter case, it is up to the producers and consumers of RIF-PRD rulesets that reference non-builtin predicates to agree on their semantics.

Example 2.7. The example below shows the RIF XML serialization of an externally defined atomic formula that tests whether the value denoted by the variable named *?a* (e.g. the age of a chicken) is greater than the integer value 8, where the test is intended to behave like the builtin predicate op:numeric-greater-than (http://www.w3.org/TR/xpath-functions/#func-numeric-greater-than) as specified in XQuery 1.0 and XPath 2.0 Functions and Operators (http://www.w3.org/TR/xpath-functions/) .

In the example, the prefix `op:` is associated with the namespace http://www.w3.org/2007/rif-builtin-predicate#.

```
<External>
   <content>
      <Atom>
         <op> <Const type="rif:iri"> op:numeric-greater-than </Const> </op>
         <args rif:ordered="yes">
            <Var> ?a </Var>
            <Const type="xsd:decimal"> 8 </Const>
         </args>
      </Atom>
   </content>
</External>
```

5.2.3 FORMULA

The FORMULA class is used to serialize condition formulas, that is, atomic formulas, conjunctions, disjunctions, negations and existentials.

As an abstract class, FORMULA is not associated with specific XML markup in RIF-PRD instance documents.

```
   [ ATOMIC | And | Or | NmNot | Exists ]
```

5.2.3.1 ATOMIC

An atomic formula is serialized using a single ATOMIC statement. See specification of ATOMIC, above.

5.2.3.2 And

A conjunction is serialized using the And element.

The And element contains zero or more formula sub-elements, each containing an element of the FORMULA group.

```
   <And>
      <formula> FORMULA </formula>*
   </And>
```

5.2.3.3 Or

A disjunction is serialized using the Or element.

The Or element contains zero or more formula sub-elements, each containing an element of the FORMULA group.

```
   <Or>
      <formula> FORMULA </formula>*
   </Or>
```

5.2.3.4 NmNot

A negation is serialized using the `NmNot` element.

The `NnNot` element contains exactly one `formula` sub-element. The `formula` element contains an element of the `FORMULA` group, that serializes the negated statement.

```
<NmNot>
   <formula> FORMULA </formula>
</NmNot>
```

Editor's Note: The name of that construct may change, including the tag of the XML element.

5.2.3.5 Exists

An existentially quantified formula is serialized using the `Exists` element.

The `Exists` element contains:

- one or more `declare` sub-elements, each containing a `Var` element that serializes one of the existentially quantified variables;
- exactly one required `formula` sub-element that contains an element from the `FORMULA` abstract class: the `FORMULA` serializes the formula in the scope of the quantifier.

```
<Exists>
   <declare> Var </declare>+
   <formula> FORMULA </formula>
</Exists>
```

Example 2.8. The example below shows the RIF XML serialization of a boolean expression that tests whether the chicken denoted by variable *?c* is older than 8 months, by testing the existence of a value, denoted by variable *?a*, that is both the age of *?c*, as serialized as a `Frame` element, as in example 2.6, and greater than 8, as serialized as an `External` `ATOMIC`, as in example 2.7.

```
<Exists>
   <declare> <Var> a </Var> </declare>
   <formula>
      <And>
         <Frame>
            <object> <Var> c </Var> </object>
            <slot rif:ordered="yes">
               <Const type="rif:iri">
                  http://example.com/2008/joe#Chicken/age
               </Const>
               <Var> a </Var>
            </slot>
         </Frame>
         <External>
            <content>
               <Atom>
                  <op> <Const type="rif:iri"> op:numeric-greater-than </Const> </op>
                  <args rif:ordered="yes">
                     <Var> a </Var>
                     <Const type="xsd:decimal"> 8 </Const>
                  </args>
               </Atom>
            </content>
         </External>
      </And>
   </formula>
</Exists>
```

5.3 Actions

This section specifies the XML syntax that is used to serialize the action part of a rule supported by RIF-PRD.

5.3.1 ATOMIC_ACTION

The `ATOMIC_ACTION` class of elements is used to serialize the atomic actions: *assert* and *retract*.

```
   [ Assert | Retract ]
```

5.3.1.1 Assert

An atomic assertion action is serialized using the `Assert` element.

An atom (positional or with named arguments), a frame, a membership atomic formula and a subclass atomic formula can be asserted.

The `Assert` element has one `target` sub-element that contains an `Atom`, a `Frame`, a `Member` or a `Subclass` element that represents the facts to be added on performing the action.

```
   <Assert>
        <target> [ Atom | Frame | Member | Subclass ] </target>
   </Assert>
```

5.3.1.2 Retract

The `Retract` construct is used to serialize retract atomic actions, that result in removing a fact from the fact base. Only atoms (positional or with named arguments), frames and frame objects can be retracted.

The `Retract` element has one `target` sub-element that contains an `Atom`, a `Frame`, or a `TERM` construct that represents the facts or the object to be removed on performing the action.

```
<Retract>
   <target> [ Atom | Frame | TERM ] </target>
</Retract>
```

Example 2.10. The example below shows the RIF XML representation of an action that updates the chicken-potato ownership table by removing the predicate that states that the chicken denoted by variable *?c* owns the potato denoted by variable *?p*. The predicate is represented as in example 2.4.

```
<Retract>
   <target>
      <Atom>
         <op>
            <Const type="rif:iri">
               http://example.com/2008/joe#owns
            </Const>
         </op>
         <args rif:ordered="yes">
            <Var> c </Var>
            <Var> p </Var>
         </args>
      </Atom>
   </target>
</Retract>
```

5.3.2 INITIALIZATION

The `INITIALIZATION` class of elements is used to serialized the constructs that specify the initial value assigned to an action variable, in an action variable declaration: it can be a new frame identifier or the slot value of a frame.

As an abstract class, `INITIALIZATION` is not associated with specific XML markup in RIF-PRD instance documents.

```
[ New | Frame ]
```

5.3.2.1 New

The `New` element is used to serialize the construct used to create a new frame identifer.

The `New` element has an `instance` sub-element that contains a `Var`, which serializes the action variable intended to be assigned the new frame identifier.

```
<New>
   <instance> Var </instance>?
</New>
```

5.3.2.2 Frame

The `Frame` element is used, with restrictions, to the serialize the construct used to assign an action variable the slot value of a frame.

In that position, a `Frame` must contain, in addition to its `object` sub-element, one and only one `slot` sub-element, that contains a sub-element of the `TERM` class, serializing the slot name, and a `Var` sub-element, that serializes the action variable intended to be assigned the slot value.

```
<Frame>
   <object> TERM </object>
   <slot rif:ordered="yes">
      TERM
      Var
   </slot>
</Frame>
```

5.3.3 ACTION_BLOCK

The `ACTION_BLOCK` class of constructs is used to represent the conclusion, or action part, of a production rule serialized using RIF-PRD.

If action variables are declared in the action part of a rule, or if some atomic actions are not assertions, the conclusion must be serialized as a full action block, using the `Do` element. However, simple action blocks that contain only one or more assert actions can be serialized like the conclusions of logic rules using RIF-Core or RIF-BLD, that is, as a single asserted `ATOMIC` or as a conjunction of the asserted `ATOMICS`.

As an abstract class, `ACTION_BLOCK` is not associated with specific XML markup in RIF-PRD instance documents.

```
[ Do | And | ATOMIC ]
```

5.3.3.1 Do

An action block is serialized using the `Do` element.

A `Do` element contains:

- zero or more `actionVar` sub-elements, each of them used to serialize one action variable declaration. Accordingly, an `actionVar` element must contain a `Var` sub-element, that serializes the declared variable; followed by a sub-element of the `INITIALIZATION` class, that serializes the initial value assigned to the declared variable;
- one `actions` sub-element that serializes the sequence of atomic actions in the action block, and that contains, accordingly, a sequence of one or more sub-elements of the `ATOMIC_ACTION` class.

```
<Do>
   <actionVar rif:ordered="yes">
      Var
      INITIALIZATION
   </actionVar>*
   <actions rif:ordered="yes">
      ATOMIC_ACTION+
   </actions>
</Do>
```

Example 2.9. The example below shows the RIF XML representation of an action block that asserts a new 100 decigram potato.

```
<Do>
   <actionVar>
      <Var>p</Var>
      <New>
         <instance><Var>p</Var></instance>
      </New>
   </actionVar>
   <actions rif:ordered="yes">
      <Assert>
         <target>
            <Member>
               <instance><Var>p</Var></instance>
               <class>
                  <Const type="rif:iri">http://example.com/2008/joe#Potato</Const>
               </class>
            </Member>
         </target>
      </Assert>
      <Assert>
         <target>
            <Frame>
               <object><Var>p</Var></object>
               <slot rif:ordered="yes">
                  <Const type="rif:iri">http://example.com/2008/joe#weight</Const>
                  <Const type="xsd:decimal">100</Const>
               </slot>
            </Frame>
         </target>
      </Assert>
   </actions>
</Do>
```

5.3.3.2 And

An action block that contains only assert atomic actions can be serialized using the `And` element, for compatibility with RIF-Core and RIF-BLD.

However, the atomic formulas allowed as conjuncts are restricted to atoms (positional or with named arguments), frames, and membership or subclass atomic formulas.

In that position, an `And` element must contain at least one sub-element.

```
<And>
   <formula> [ Atom | Frame | Member | Subclass ] </formula>+
</And>
```

5.3.3.3 ATOMIC

For compatibility with RIF-Core and RIF-BLD, an action block that contains only a single assert atomic action can be serialzed as the ATOMIC that serializes the target of the assert action.

However, the only atomic formulas allowed in that position are the ones that are allowed as targets to an atomic assert action: atoms (positional or with named arguments), frames, and membership or subclass atomic formulas.

```
[ Atom | Frame | Member | Subclass ]
```

5.4 Rules and Groups

This section specifies the XML constructs that are used, in RIR-PRD, to serialize rules and groups.

5.4.1 RULE

In RIF-PRD, the RULE class of constructs is used to serialize rules, that is, unconditional as well as conditional actions, or rules with bound variables.

As an abstract class, RULE is not associated with specific XML markup in RIF-PRD instance documents.

```
[ Implies | Forall | ACTION_BLOCK ]
```

5.4.1.1 ACTION_BLOCK

An unconditional action block is serialized, in RIF-PRD XML, using the ACTION_BLOCK class of construct.

5.4.1.2 Implies

Conditional actions are serialized, in RIF-PRD, using the XML element Implies.

The Implies element contains an optional if sub-element and a then sub-element:

- the optional if element contains an element from the FORMULA class of constructs, that serializes the condition of the rule;
- the required then element contains one element from the ACTION_BLOCK class of constructs, that serializes its conlusion.

```
<Implies>
   <if> FORMULA </if>?
   <then> ACTION_BLOCK </then>
</Implies>
```

5.4.1.3 Forall

The Forall construct is used, in RIF-PRD, to represent rules with bound variables.

The `Forall` element contains:

- one or more `declare` sub-elements, each containing a `Var` element that represents one of the universally quantified variables;
- zero or more `pattern` sub-elements, each containing an element from the `FORMULA` group of constructs, serializing one binding pattern;
- exactly one `formula` sub-element that serializes the formula in the scope of the variables binding, and that contains an element of the `RULE` group.

```
<Forall>
   <declare> Var </declare>+
   <pattern> FORMULA </pattern>*
   <formula> RULE </formula>
</Forall>
```

Editor's Note: Nested `Forall`s make explicit the scope of the declared variables, and, thus, impose an order on the evaluation of the `pattern` and `if` `FORMULA`s in a rule. That ordering and the use of patterns to constrain the binding of variables may be of practical significance for some production rule systems, but they are irrelevant with respect to the intended semantics of the rules being interchanged (although they would be relevant if RIF-PRD was to be extended to support some kind of "else" or "case" construct). In addition, RIF-BLD does not allow nested `Forall` and does not support the association of contraining `pattern`s to declared variables. The working group seeks feedback regarding whether nested `Forall` and constrainting `pattern` should be supported in RIF-PRD, to the cost of reducing the interoperability with RIF-BLD (http://www.w3.org/2005/rules/wiki/) .

Example 2.10. The example below shows how the CMP rule extract: *"if a chicken owns a potato and ..."* could be serialized using a binding `pattern` `FORMULA`:

```
<Forall>
   <declare><Var>chicken</Var></declare>
   <formula>
      <Forall>
         <declare><Var>potato</Var></declare>
         <pattern>
            <External>
               <content>
                  <Atom>
                     <Const type="rif:iri">
                        http://example.com/2008/joe#owns
                     </Const>
                     <Args rif:ordered="yes">
                        <Var>chicken</Var>
                        <Var>potato</Var>
                     </Args>
                  </Atom>
               </content>
            </External>
         </pattern>
         <formula>
            ...
         </formula>
      </Forall>
   </formula>
</Forall>
```

5.4.2 Group

The `Group` construct is used to serialize a group.

The `Group` element has zero or one `behavior` sub-element and zero or more `sentence` sub-elements:

- the `behavior` element contains
  - zero or one `ConflictResolution` sub-element that contains exactly one IRI. The IRI identifies the conflict resolution strategy that is associated with the `Group`; and
  - zero or one `Priority` sub-element that contains exactly one signed integer between -10,000 and 10,000. The integer associates a priority with the `Group`'s sentences;
- a `sentence` element contains either a `Group` element or an element of the `RULE` abstract class of constructs.

```
<Group>
   <behavior>
      <ConflictResolution> xsd:anyURI </ConflictResolution>?
      <Priority> -10,000 ≤ xsd:int ≤ 10,000 </Priority>?
   </behavior>?
   <sentence> [ RULE | Group ] </sentence>*
</Group>
```

5.5 Constructs carrying no semantics

5.5.1 Document

This does have semantics

The `Document` is the root element in a RIF-PRD instance document.

The `Document` contains zero or one `payload` sub-element, that must contain a `Group` element.

```
<Document>
   <payload> Group </payload>?
</Document>
```

5.5.2 ~~Metadata~~ Annotations

refer to BLD

Metadata can be associated with any concrete class element in RIF-PRD: those are the elements with a CamelCase tagname starting with an upper-case character:

```
CLASSELT = [ TERM | ATOMIC | FORMULA | ACTION | RULE | Group | Document ]
```

An identifier can be associated to any instance element of the abstract `CLASSELT` class of constructs, as an optional `id` sub-element that MUST contain a `Const` of type `rif:local` or `rif:iri`.

Metadata can be included in any instance of a concrete class element using the `meta` sub-element.

The RIF-PRD `Frame` construct is used to serialize metadata: the content of the `Frame`'s `object` sub-element identifies the object to which the metadata is associated:, and the `Frame`'s `slots` represent the metadata properly said as property-value pairs.

If the all the metadata is related to the same object, the `meta` element can contain a single `Frame`

sub-element. If metadata related to several different objects need be serialized, the `meta` role element can contain an `And` element with zero or more `formula` sub-elements, each containing one `Frame` element.

```
<CLASSELT>
    <id> Const </id>?
    <meta>
       [ Frame
         |
         <And>
            <formula> Frame </formula>*
         </And>
       ]
    </meta>?
    other CLASSELT content
 </CLASSELT>
```

ABSTRACT ?!

Notice that the content of the `meta` sub-element of an instance of a RIF-PRD class element is not necessarily associated to that same instance element: only the content of the `object` sub-element of the `Frame` that represents the metadata specifies what the metadata is about, not where it is included in the instance RIF document.

It is suggested to use Dublin Core, RDFS, and OWL properties for metadata, along the lines of http://www.w3.org/TR/owl-ref/#Annotations -- specifically owl:versionInfo, rdfs:label, rdfs:comment, rdfs:seeAlso, rdfs:isDefinedBy, dc:creator, dc:description, dc:date, and foaf:maker.

Example 2.11. TBC

6 Presentation syntax

earlier ?

PRD

To make it easier to read, a non-normative, lightweight notation was introduced to complement the mathematical english specification of the abstract syntax and the semantics of RIF-PRD. This section specifies a presentation syntax for RIF-PRD, that extends that notation. The presentation syntax is not normative. However, it may help implementers by providing a more succinct overview of RIF-PRD syntax.

Editor's Note: An uptodate version of the RIF-PRD presentation synatx will be included in a future version of this document.

7 References

[CIR04]
Production Systems and Rete Algorithm Formalisation (http://hal.inria.fr/docs/00/28/09/38/PDF /rete.formalisation.pdf) , Cirstea H., Kirchner C., Moossen M., Moreau P.-E. Rapport de recherche n° inria-00280938 - version 1 (2004).

[CURIE]
CURIE Syntax 1.0 - A compact syntax for expressing URIs (http://www.w3.org/2001/sw /BestPractices/HTML/2005-10-27-CURIE) , W3C note 27 October 2005, M. Birbeck (ed.).

[HAK07]
Data Models as Constraint Systems: A Key to the Semantic Web (http://www.cs.brown.edu/people/pvh/CPL/Papers/v1/hak.pdf) , Hassan Ait-Kaci, Constraint Programming Letters, 1:33--88, 2007.

[PLO04]
A Structural Approach to Operational Semantics (http://homepages.inf.ed.ac.uk/gdp/publications/sos_jlap.pdf) , Gordon D. Plotkin, Journal of Logic and Algebraic Programming, Volumes 60-61, Pages 17-139 (July - December 2004).

[PRR07]
Production Rule Representation (PRR) (http://www.omg.org/spec/PRR/1.0/Beta1/) , OMG specification, version 1.0, 2007.

[RDF-CONCEPTS]
Resource Description Framework (RDF): Concepts and Abstract Syntax, Klyne G., Carroll J. (Editors), W3C Recommendation, 10 February 2004, http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/. Latest version available at http://www.w3.org/TR/rdf-concepts/.

[RDF-SCHEMA]
RDF Vocabulary Description Language 1.0: RDF Schema, Brian McBride, Editor, W3C Recommendation 10 February 2004, http://www.w3.org/TR/2004/REC-rdf-schema-20040210/. Latest version available at http://www.w3.org/TR/rdf-schema/.

[RFC-3066]
RFC 3066 - Tags for the Identification of Languages, H. Alvestrand, IETF, January 2001, http://www.isi.edu/in-notes/rfc3066.txt.

[RFC-3987]
RFC 3987 - Internationalized Resource Identifiers (IRIs), M. Duerst and M. Suignard, IETF, January 2005, http://www.ietf.org/rfc/rfc3987.txt.

[RIF-BLD]
RIF basic logic dialec, Boley H. and Kifer M. (Editors), W3C Rule Interchange Format Working Group Draft. Latest Version available at http://www.w3.org/2005/rules/wiki/BLD.

[RIF-Core]
RIF Core, Harold Boley, Gary Hallmark, Michael Kifer, Adrian Paschke, Axel Polleres and Dave Reynolds (Editors), W3C Rule Interchange Format Working Group Draft. Latest Version available at http://www.w3.org/2005/rules/wiki/Core.

[RIF-DTB]
RIF Datatypes and Built-Ins 1.0, Polleres A., Boley H. and Kifer M. (Editors), W3C Rule Interchange Format Working Group Draft. Latest Version available at http://www.w3.org/2005/rules/wiki/DTB.

[XDM]
XQuery 1.0 and XPath 2.0 Data Model (XDM), W3C Recommendation, World Wide Web Consortium, 23 January 2007. This version is http://www.w3.org/TR/2007/REC-xpath-datamodel-20070123/. Latest version available at http://www.w3.org/TR/xpath-datamodel/.

[XML-SCHEMA2]

XML Schema Part 2: Datatypes Second Edition, W3C Recommendation, World Wide Web Consortium, 28 October 2004, http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/. Latest version available at http://www.w3.org/TR/xmlschema-2/.

[XPath-Functions]
XQuery 1.0 and XPath 2.0 Functions and Operators, W3C Recommendation, World Wide Web Consortium, 23 January 2007, http://www.w3.org/TR/2007/REC-xpath-functions-20070123/. Latest version available at http://www.w3.org/TR/xpath-functions/.

8 Appendix: XML schema

TBD

9 Appendix: Compatibility with RIF-BLD

9.1 Syntactic compatibility between RIF-PRD and RIF-BLD

Refer to earlier sections.

Editor's Note: RIF-PRD and RIF-BLD [RIF-BLD]] share essentially the same presentation syntax and XML syntax. Future versions of this, or another, RIF document will include a complete, construct by construct, comparison table of RIF-PRD and RIF-BLD presentation and XML syntaxes.

9.2 Semantic compatibility between RIF-PRD and RIF-BLD

Refer to earlier sections.

The intended semantics of any RIF XML document which is both a syntactically valid RIF-PRD document and a syntactically valid RIF-BLD document is the same whether it is considered a RIF-PRD or a RIF-BLD document. For any input set of facts, the set of rules contained in the document must produce the same output set of facts whether it is consumed as a RIF-PRD or a RIF-BLD document.

Proof. TBC