W3C RIF in a clamshell

Christian de Sainte Marie
ILOG, an IBM company

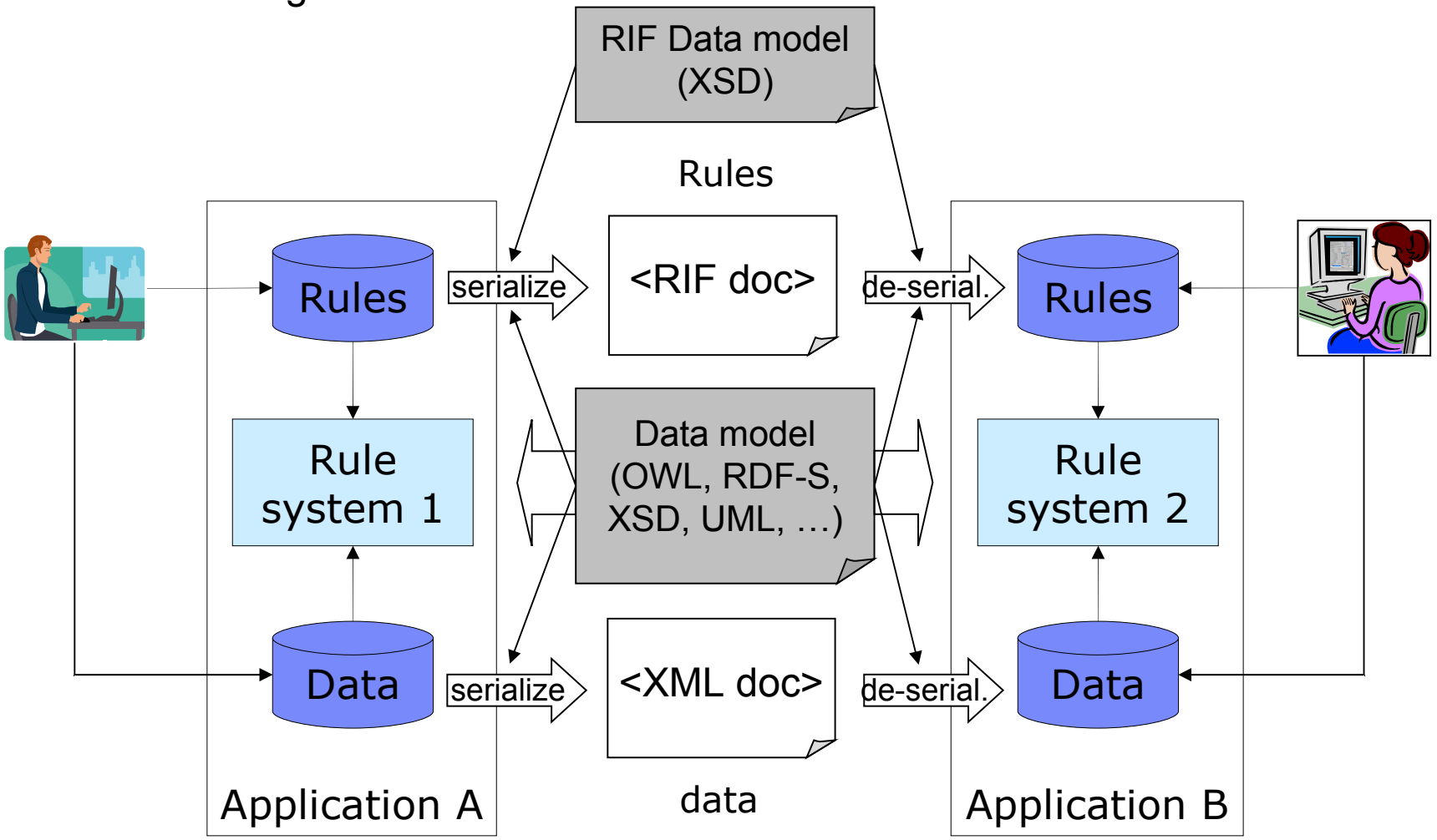
W3C RIF WG published documents

- REC track documents
  - rdf:plainLiteral (common with OWL WG)
  - RIF-Core: RIF Core dialect
  - RIF-BLD: RIF basic logic dialect
  - RIF-FLD: RIF framework for logic dialects
  - RIF-PRD: RIF production rule dialect
  - RIF-DTB: RIF data types and builtins
  - RIF-RDF-OWL: RIF RDF and OWL compatibility
  - RIF-XML-data: RIF combination with XML data and schema
- Technical reports
  - RIF-UCR: RIF use cases and requirements
  - RIF-Tests: RIF Test Cases
  - RIF-Overview: Overview of RIF documentation
  - RIF-OWL2-RL: RIF implementation of OWL2 RL

Web site: http://www.w3.org/2005/rules/wiki/RIF_Working_Group

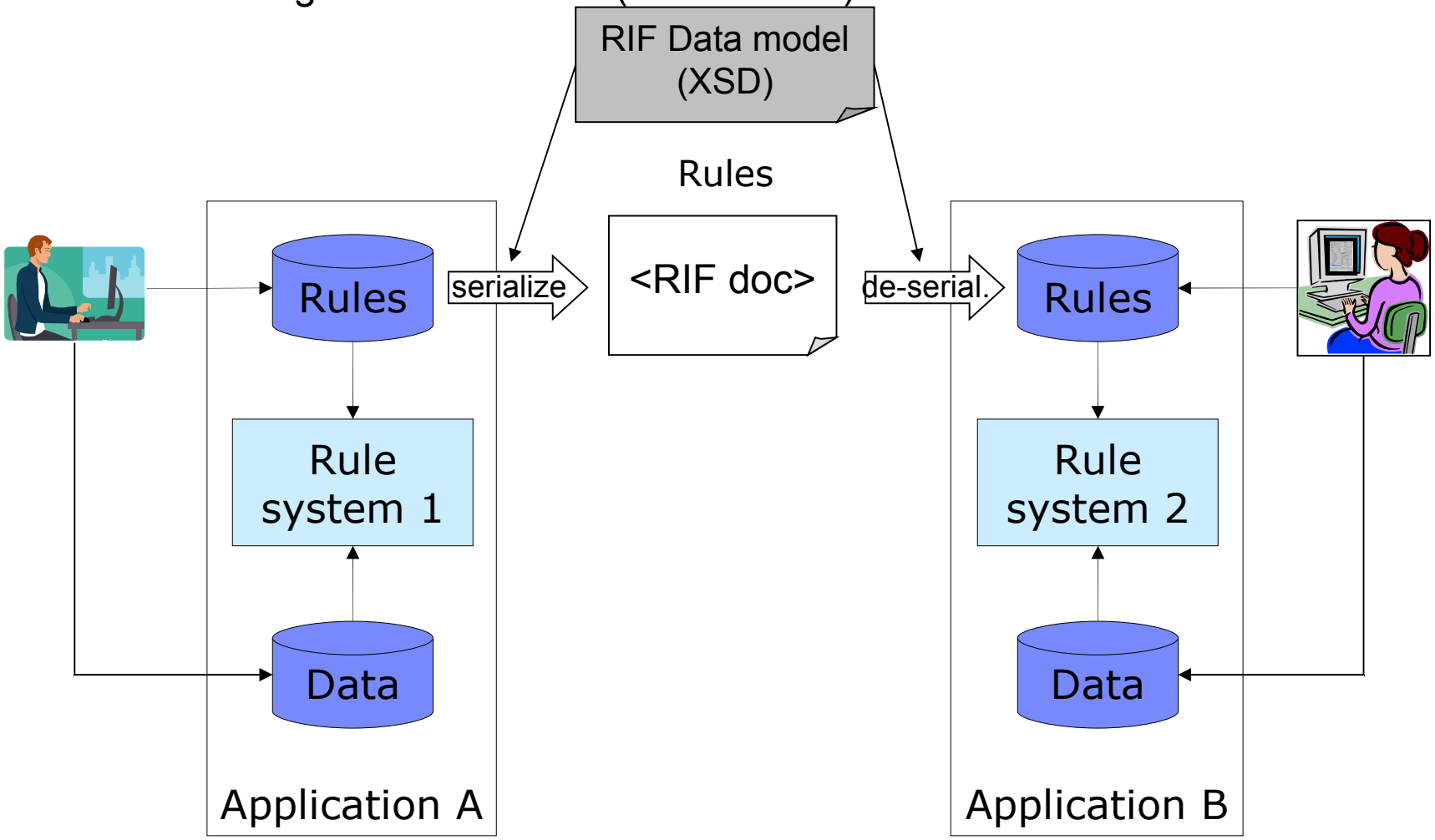
Rule interchange

Case 1: interchange rules and data



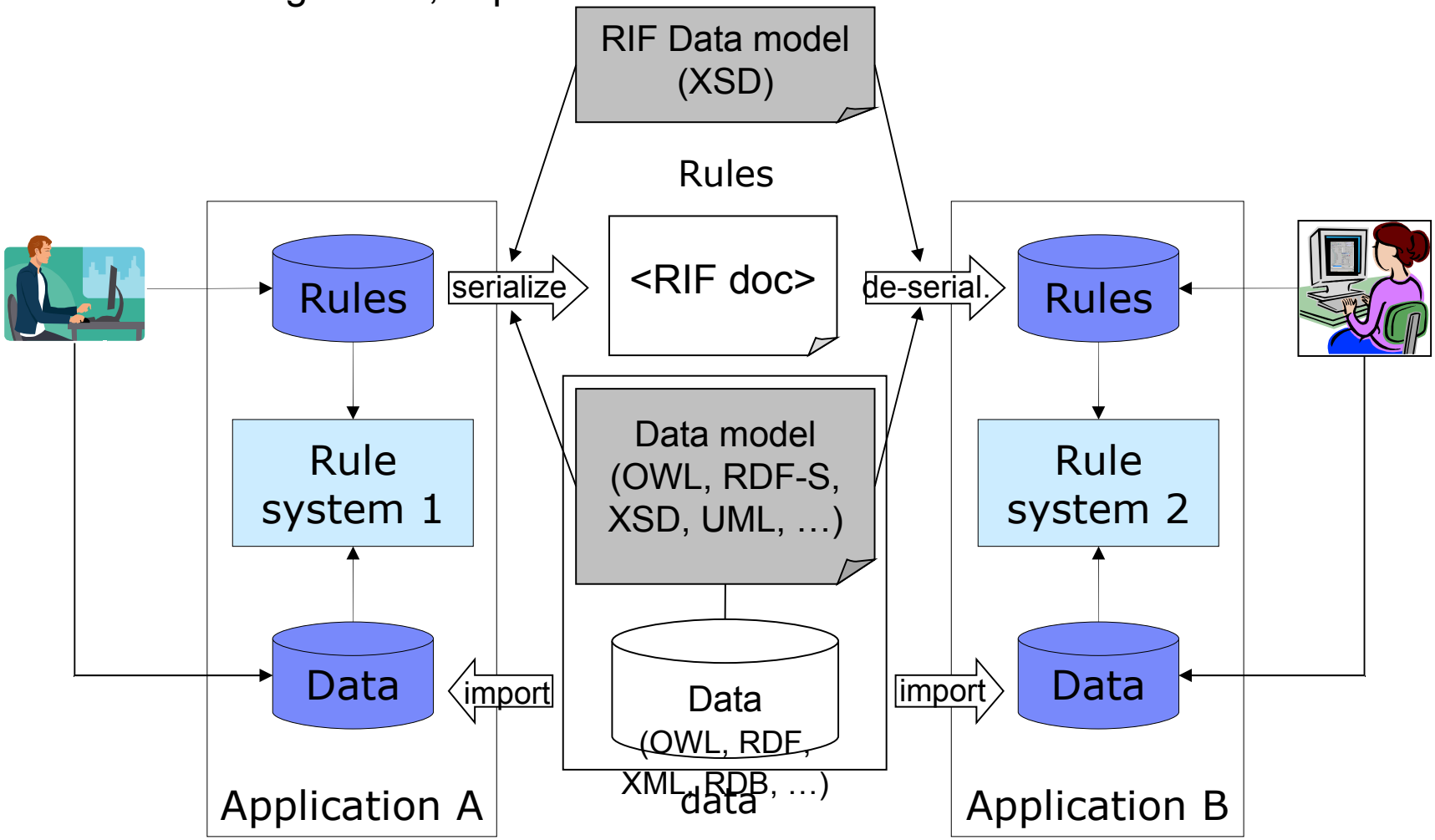
Rule interchange

Case 1': interchange rules and data (as RIF facts)



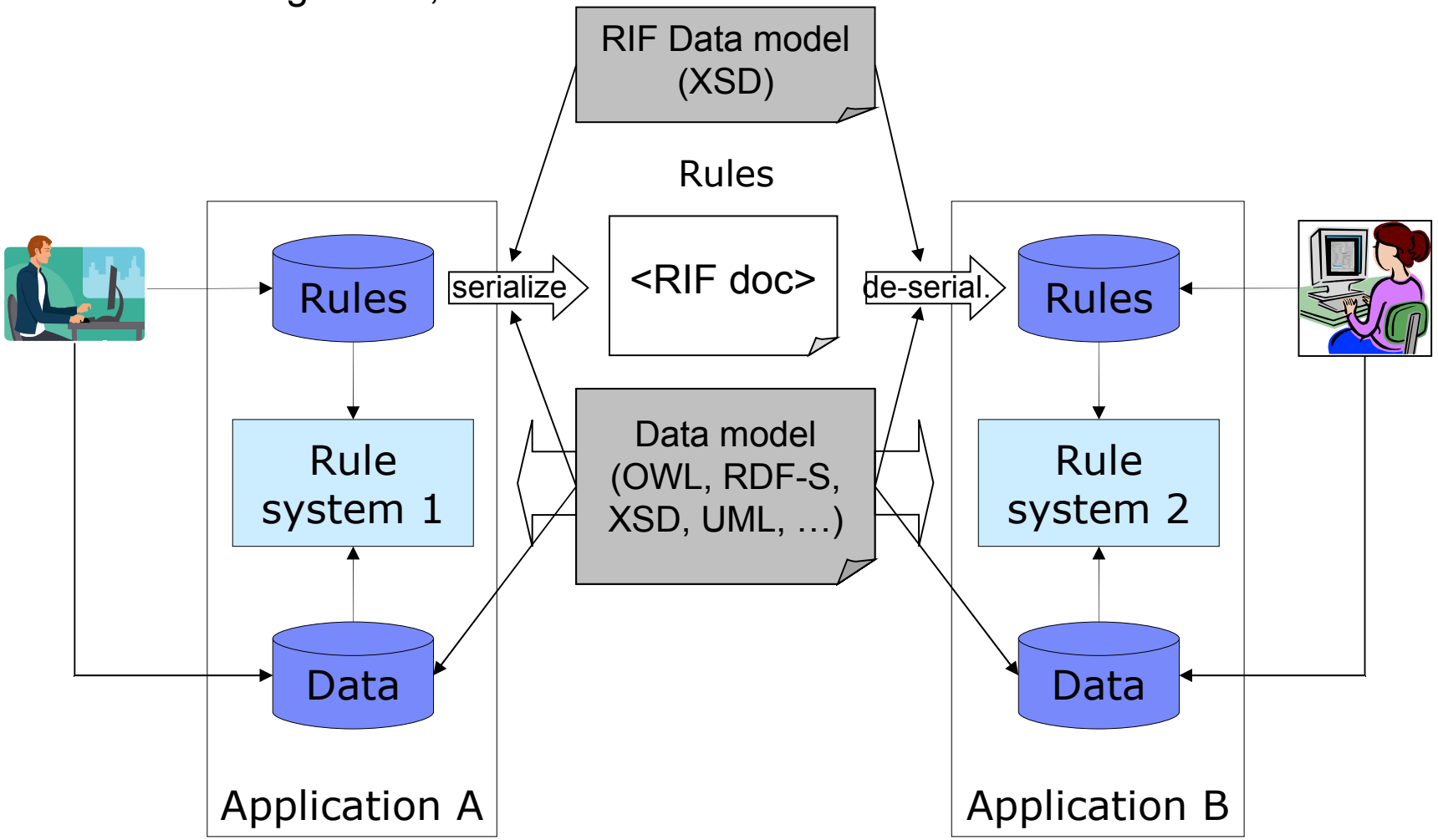
Rule interchange

Case 2: interchange rules, import data



Rule interchange

Case 3: interchange rules, use own data



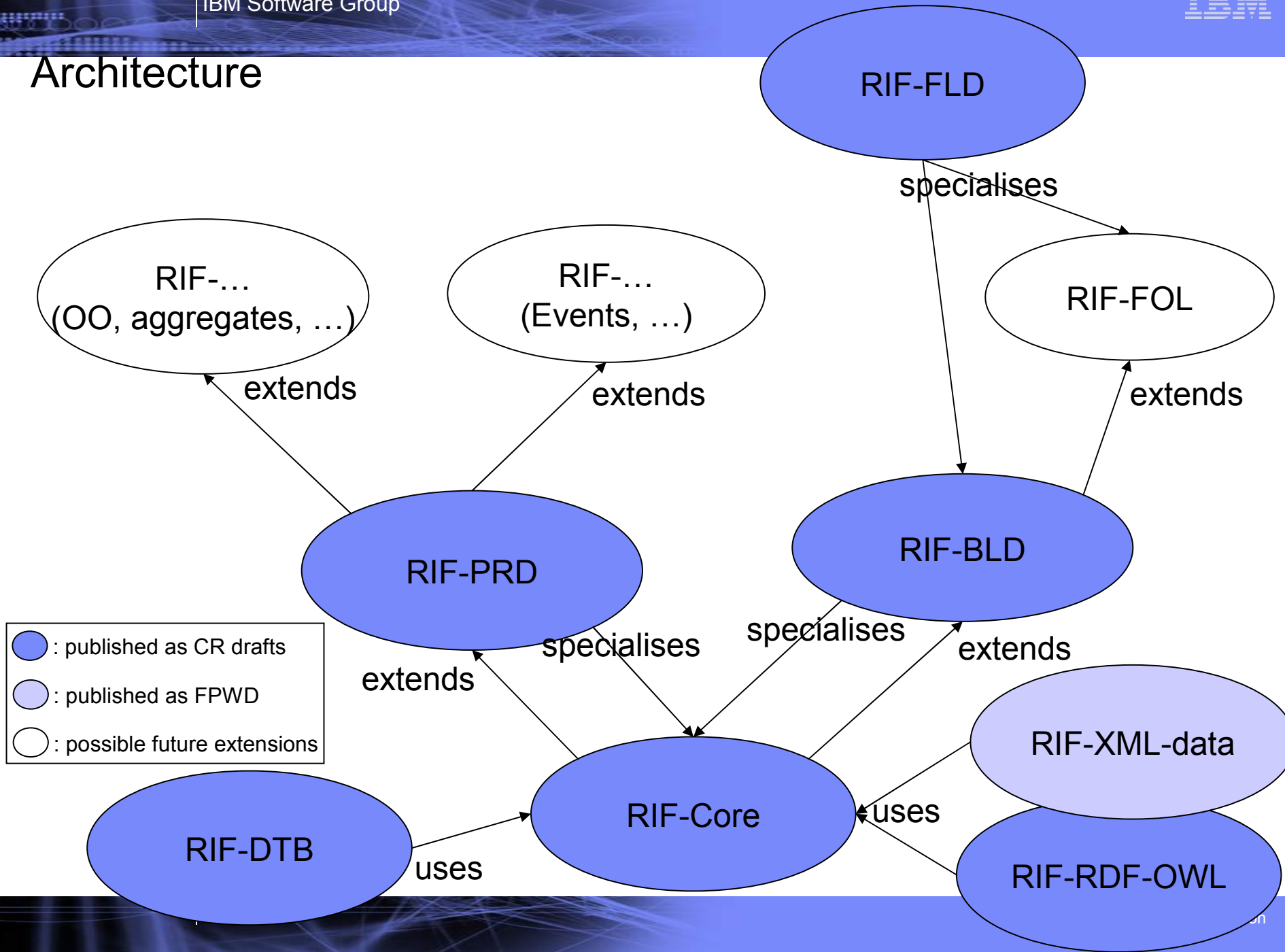
What is the Rule Interchange Format?

- Format for interchanging rules, so they can be used across diverse systems
  - allowing rules written for one application to be published, shared, and re-used in other applications and other rule engines.
  - In a semantic preserving way (between languages with compatible semantics)
  - Encouraging interoperability
  - XML syntax
  - Compatible with relevant standards (PRR, RDF, OWL, …)
- A rule is (just another) data item
  - RIF provides a standard means to feed rules into an application (at run time)
  - Semantics to prescribe (intended) application's behaviour

Design principles

- Translation paradigm
  - No intrusion in covered rule languages and rule sets
- Same semantics ⇔ same syntax
  - Share constructs accross dialects wherever they agree on the semantics
  - Different constructs where semantics do not agree
- Alternating normal form XML
  - alternating <Class> and <role> tags
  - Metadata can be attached to any class element
  - Except…
- Only XML schema is normative
  - Presentation syntax for specification's readability (examples, semantics etc)

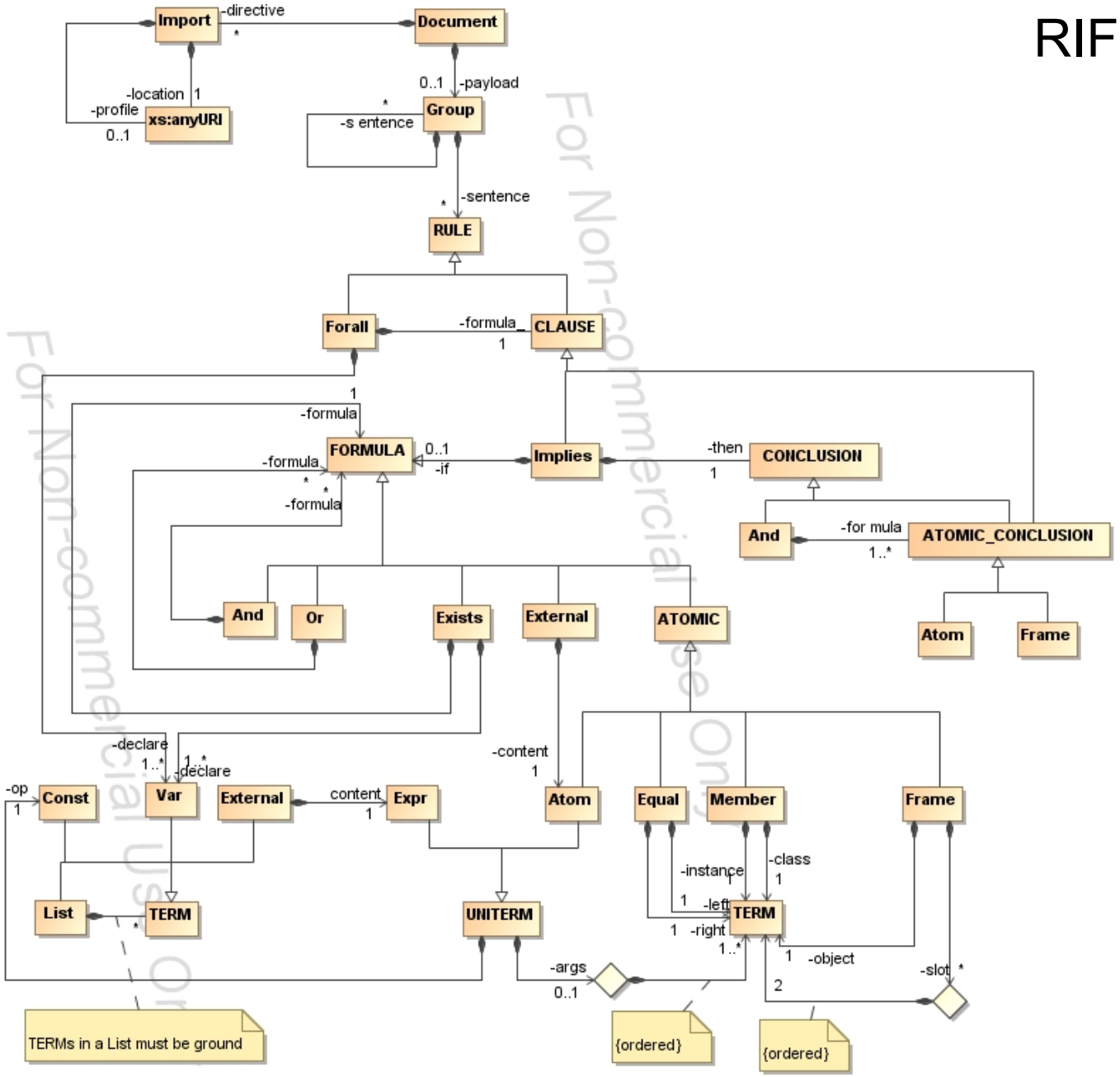
Architecture



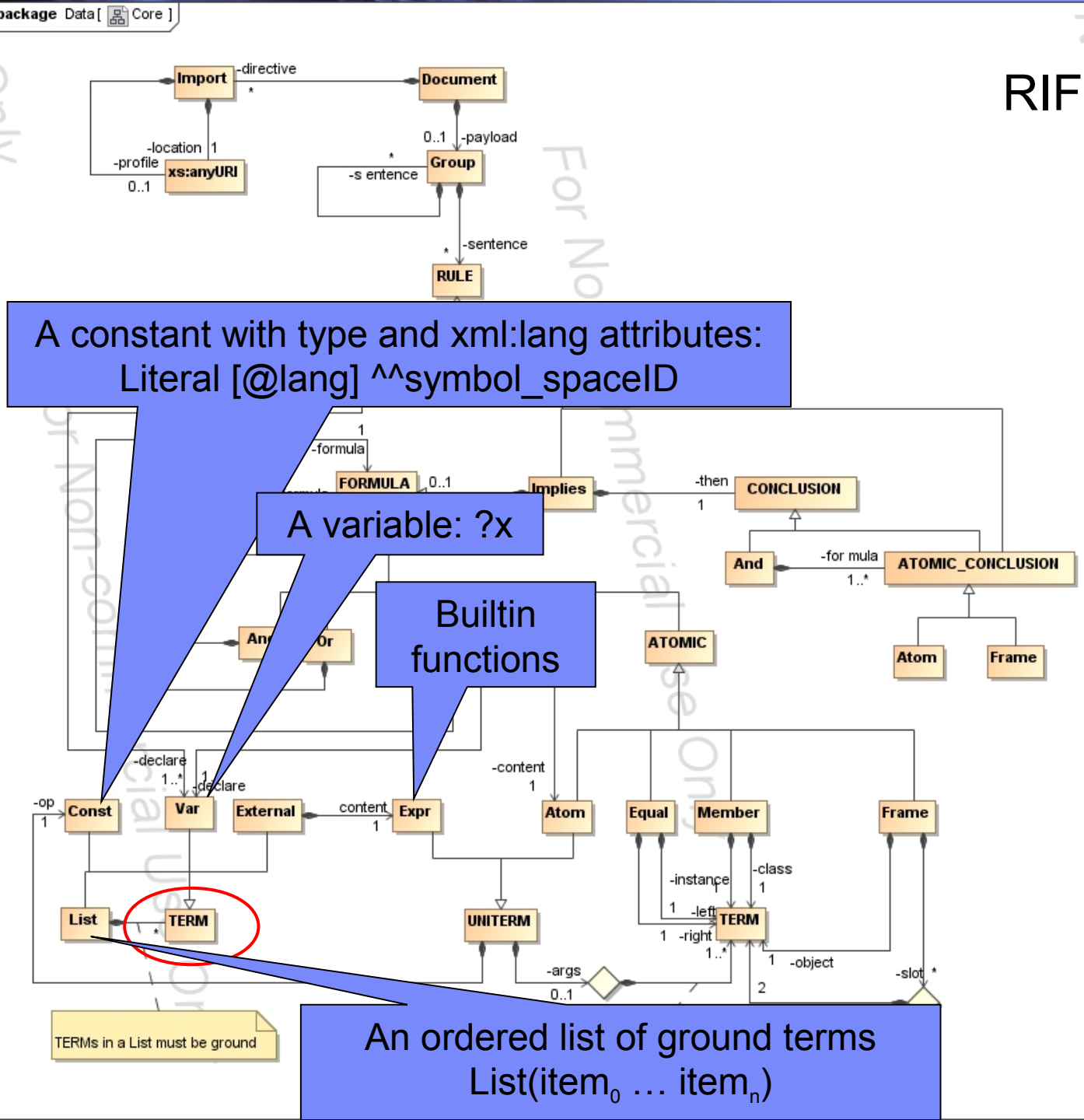
Syntax

RIF Core

RIF-Core



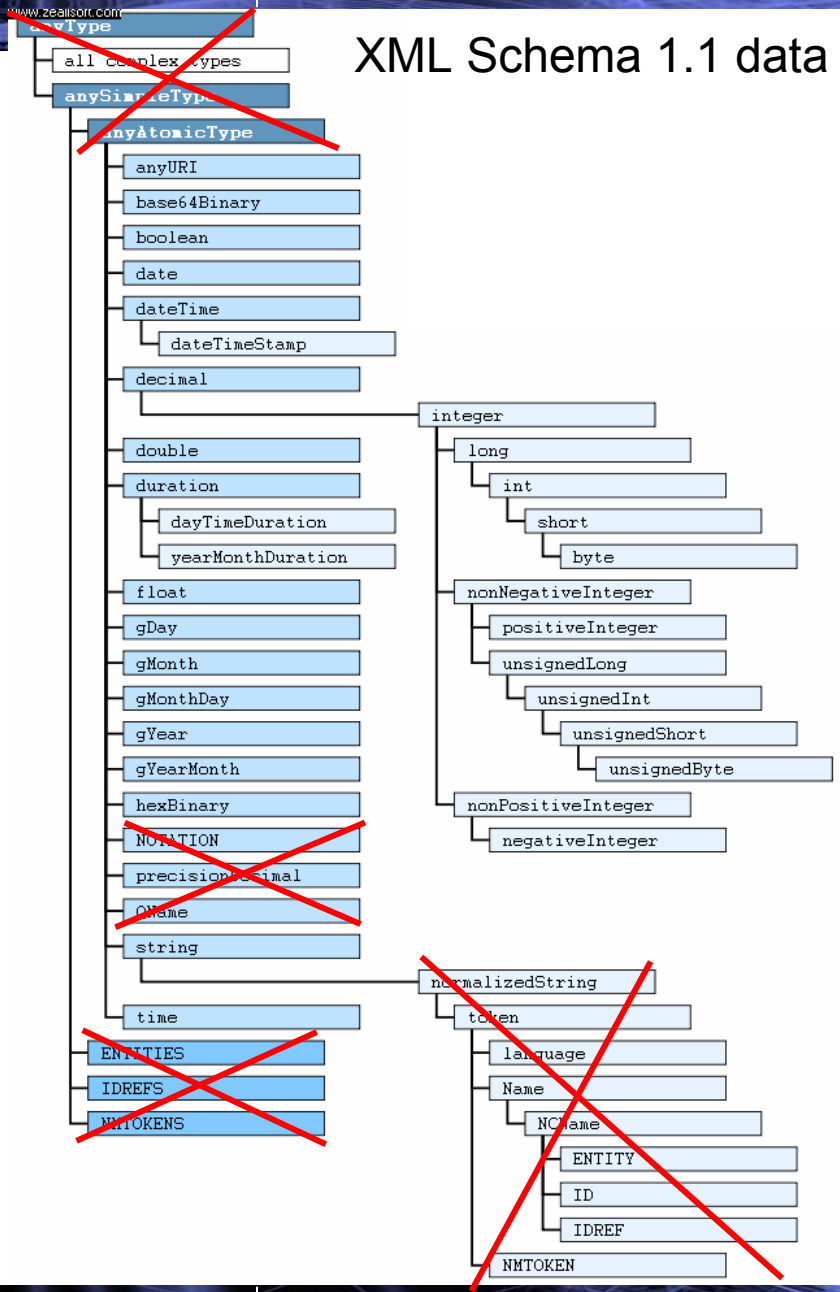
RIF-Core



Built-in data types and symbol spaces

- 33 data types from XML schema 1.1
  - All the primitive data types in XML schema minus
    - xs:NOTATION
    - xs:precisionDecimal
    - xs:QName
  - All the derived built-in data types, except those derived from xs:string
    - including xs:dayTimeDuration and xs:yearMonthDuration from the XPath 2.0 and XQuery 1.0 data model (subtypes of xs:duration)
- 2 data types in the RDF namespace
  - rdf:XMLLiteral from RDF Concepts
  - rdf:plainLiteral (from rdf:plainLiteral)
- 2 symbol spaces specific to RIF
  - A datatype is a symbol space that has
    - an associated set, called the value space, and
    - a mapping from the lexical space of the symbol space to the value space, called lexical-to-value-space mapping
  - rif:iri and rif:local have no value spaces

XML Schema 1.1 data types



+

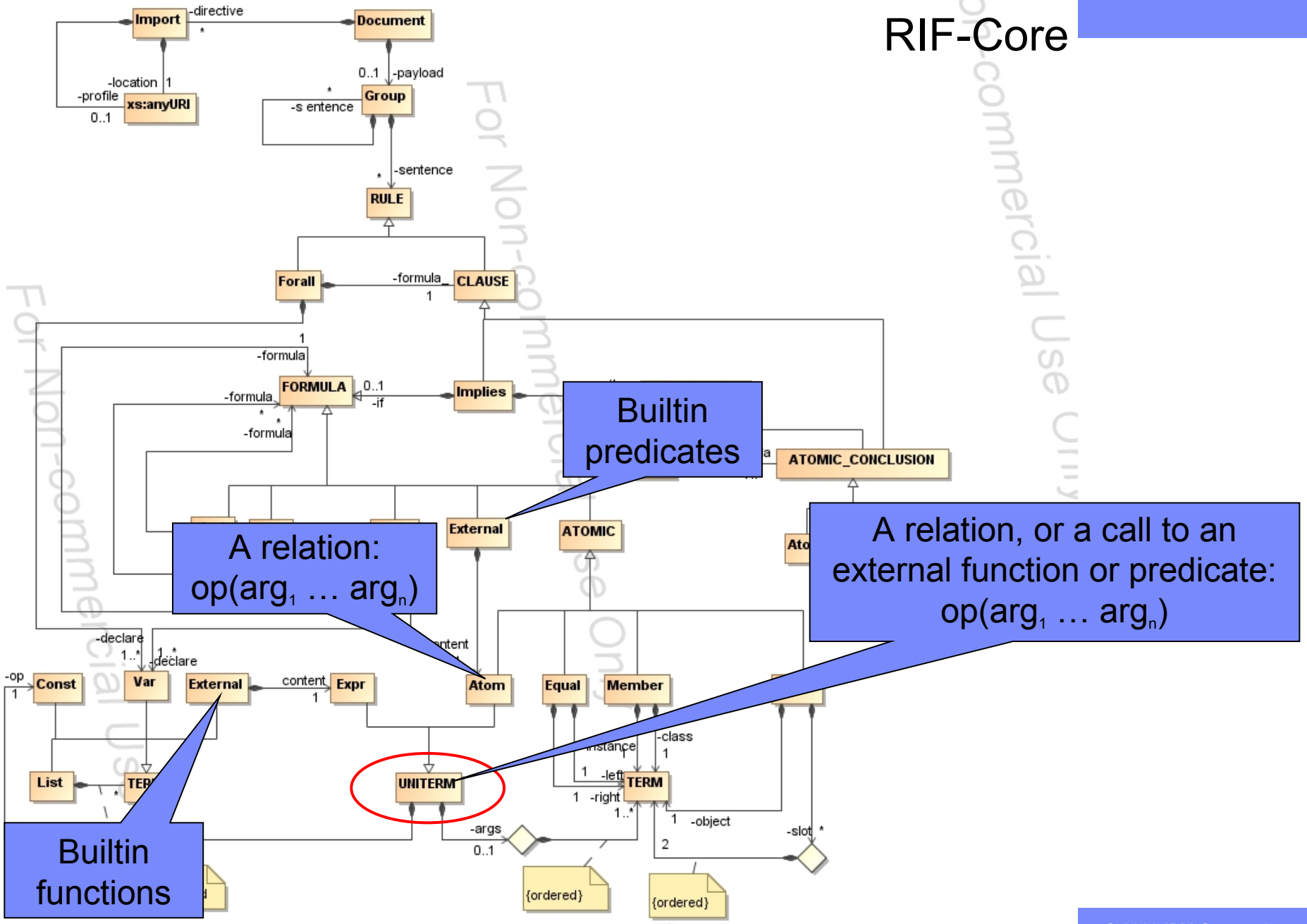
rdf:XMLLiteral

rdf:plainLiteral

rif:iri

rif:local

RIF-Core

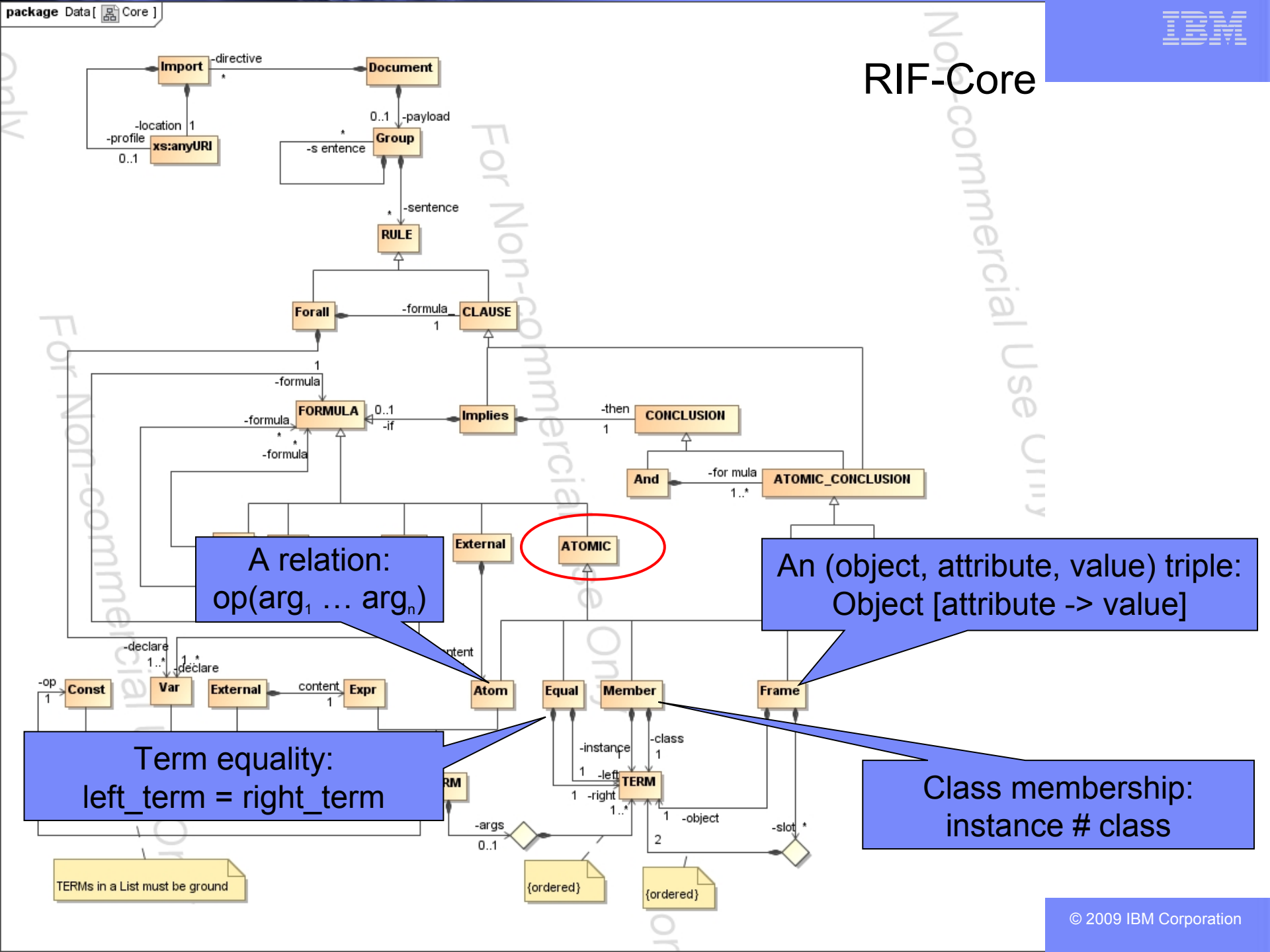


Built-in functions and predicates

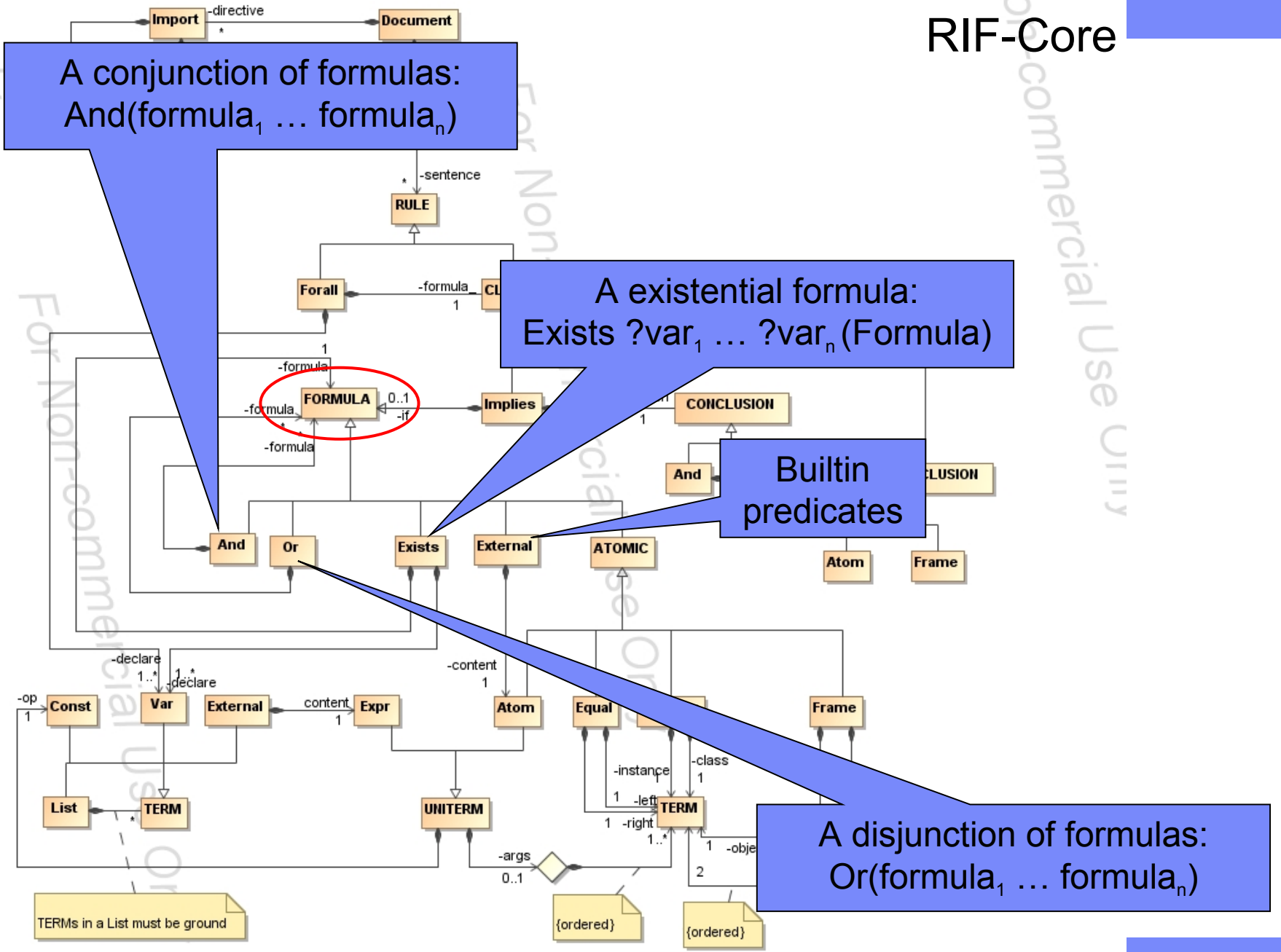
- Data type
  - Comparison: `pred:literal-not-identical`
  - Guards
    - `pred:is-literal-`*`DATATYPE`*
    - `pred:is-literal-not-`*`DATATYPE`*
  - Convertion and casting
- Numeric
  - Comparison predicates: =, !=, >, =>, <, =<
  - Binary operators: +, -, *, /, integer /, modulo
- Boolean
  - Inverse function: `func:not`
  - Comparison predicates: =, <, >
- Strings
  - 13 functions: compare, concat, joint, substring (incl. before, after), length, replace, …
  - 4 predicates: `pred:contains, pred:ends-with, pred:starts-with, pred:matches`
- Date, time and duration
  - 44 functions: extractors and arithmetic operators
  - 28 comparaison predicates
- rdf:XMLLiteral: `pred:XMLLiteral-equal, predXMLLiteral-not-equal`
- rdf:plainLiteral
  - Function: from-string, to-string, compare, length, lang
  - Predicates: `pred:matches-language-range`
- Lists
  - **At risk**: position numbered from 0
  - Predicates: `pred:is-list, pred:list-contains`
  - 14 functions: make-list, count, get, sublist, append, concatenate, insert-before, remove, reverse, index-of, union, distinct-values, intersect, except

- RIF built-in functions and predicates are *external* in the sense that their semantics is defined outside of – and independently from – the rules where they are used
  - As opposed to logic functions and predicates
- Many RIF built-in functions and predicates are adapted from *XQuery 1.0 and XPath 2.0 Functions and Operators*
  - The differences from the original XF&O include the handling of errors, the differentiation between predicates and functions, and a few other specific differences
  - If an argument value is outside of its domain, the value of the function or predicate is left unspecified
- Dialect-specific built-in functions
  - act:print

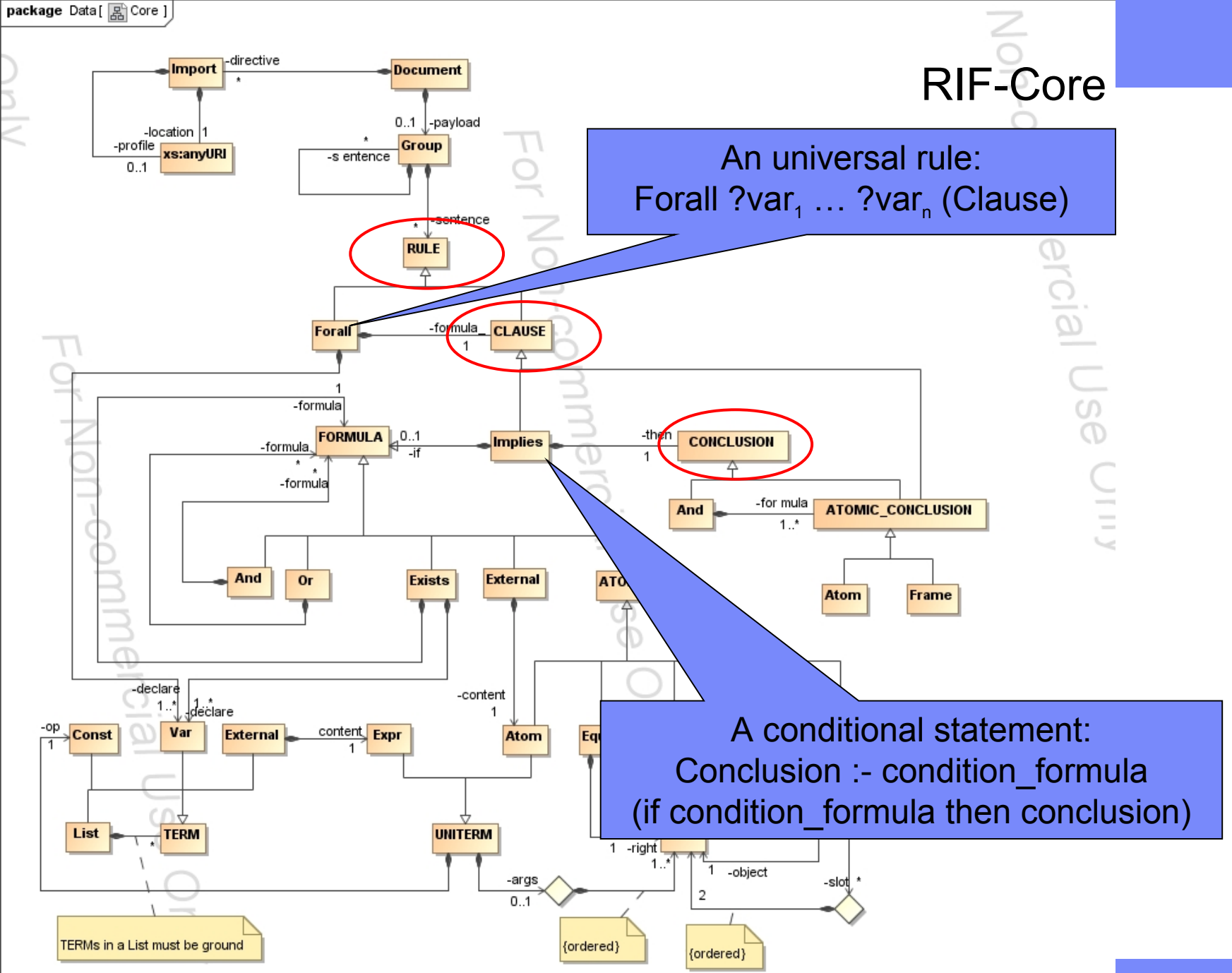
RIF-Core



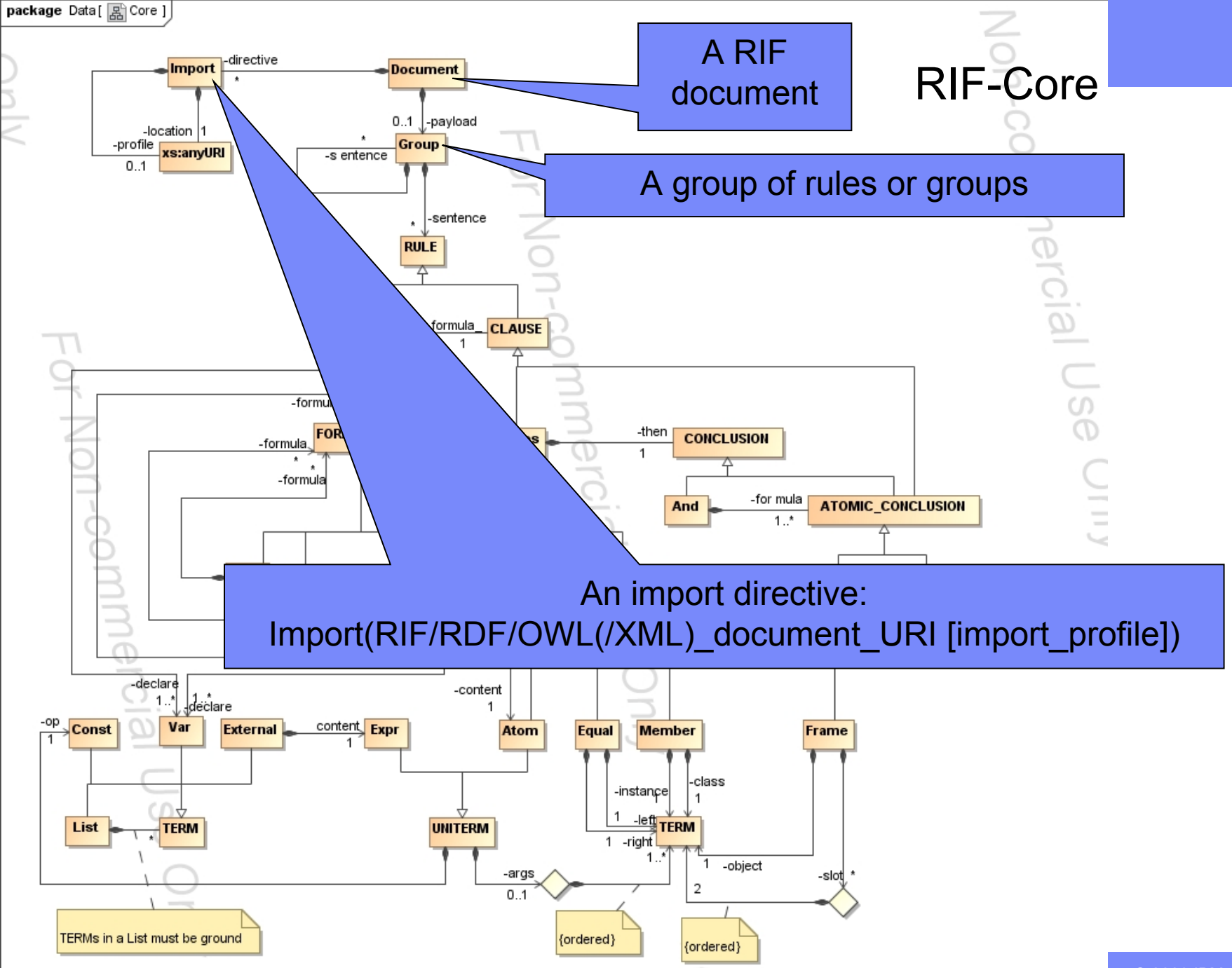
RIF-Core



RIF-Core



RIF-Core

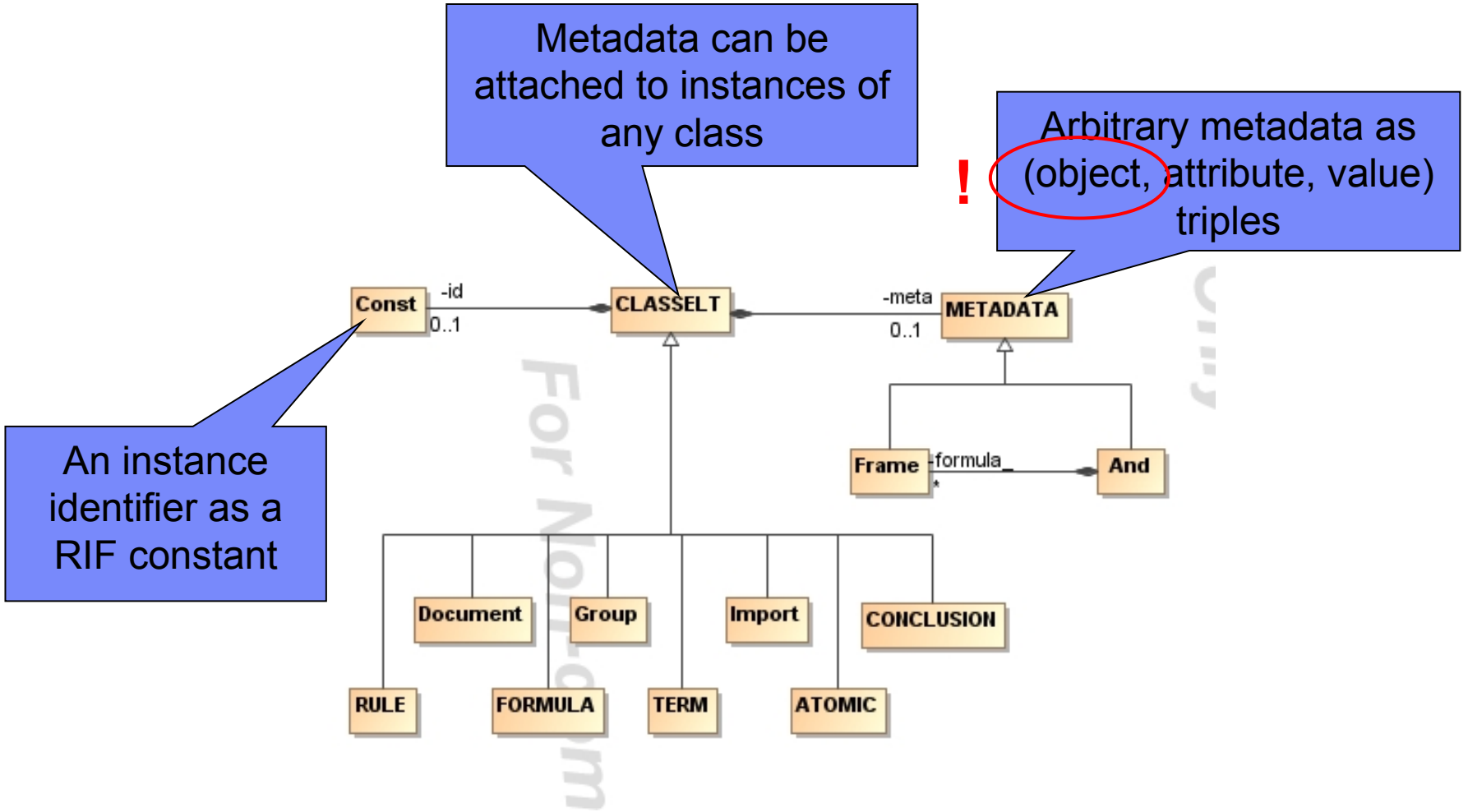


A RIF document

A group of rules or groups

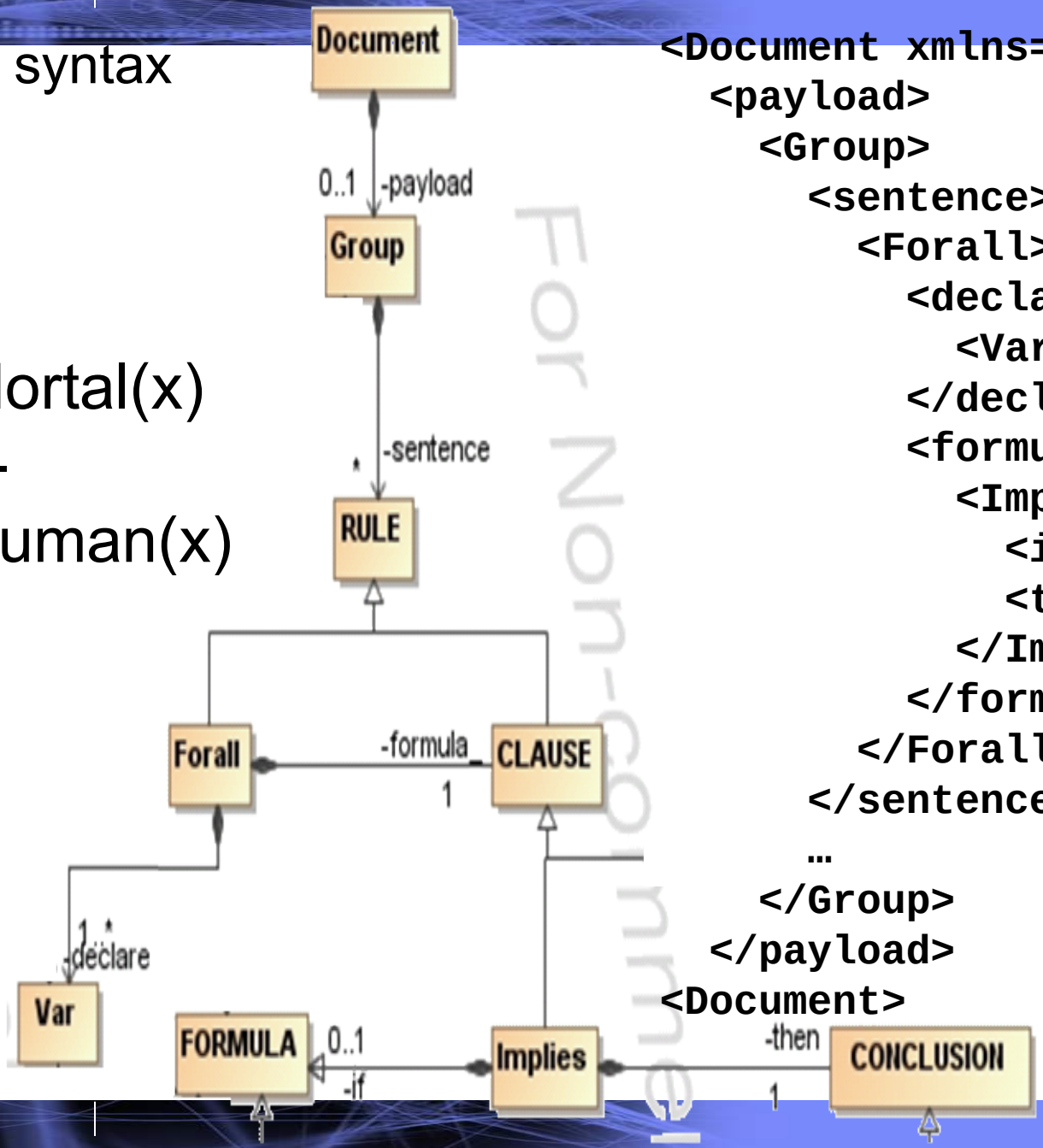
An import directive:
Import(RIF/RDF/OWL(/XML)_document_URI [import_profile])

Metadata



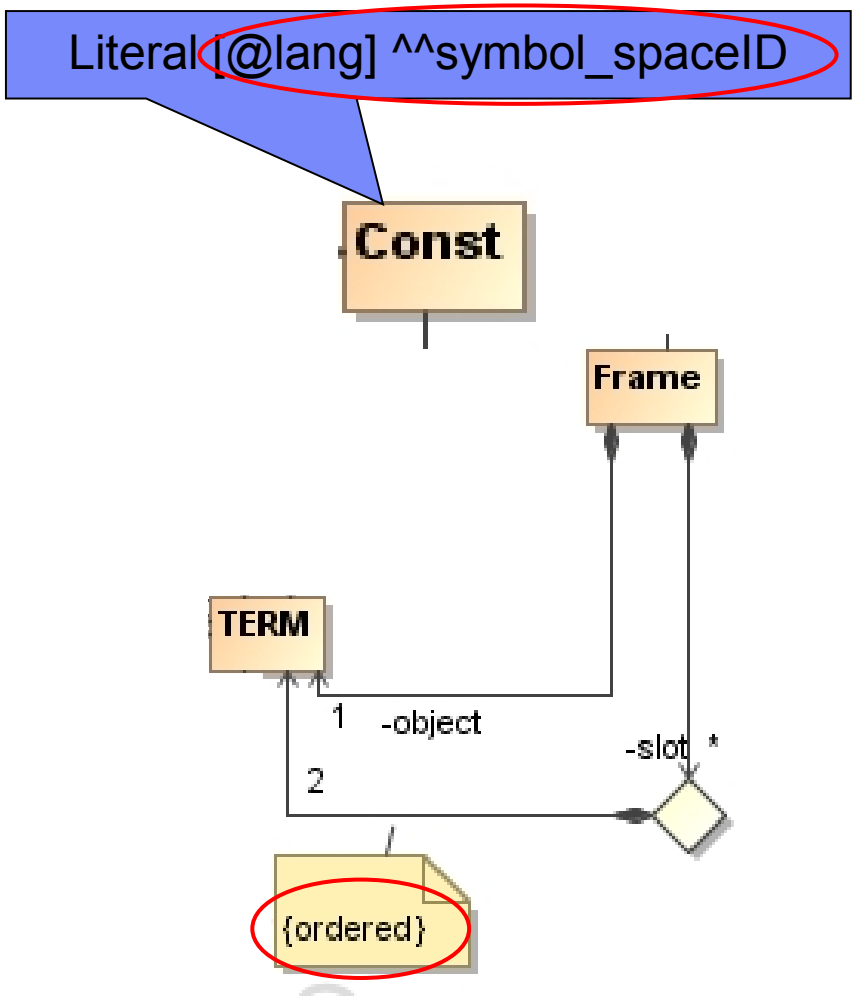
XML syntax

Mortal(x)
:-
Human(x)




```
<Document xmlns=…>
  <payload>
    <Group>
      <sentence>
        <Forall>
          <declare>
            <Var>X</Var>
          </declare>
          <formula>
            <Implies>
              <if>…</if>
              <then>…</then>
            </Implies>
          </formula>
        </Forall>
      </sentence>
      …
    </Group>
  </payload>
<Document>
```

XML syntax



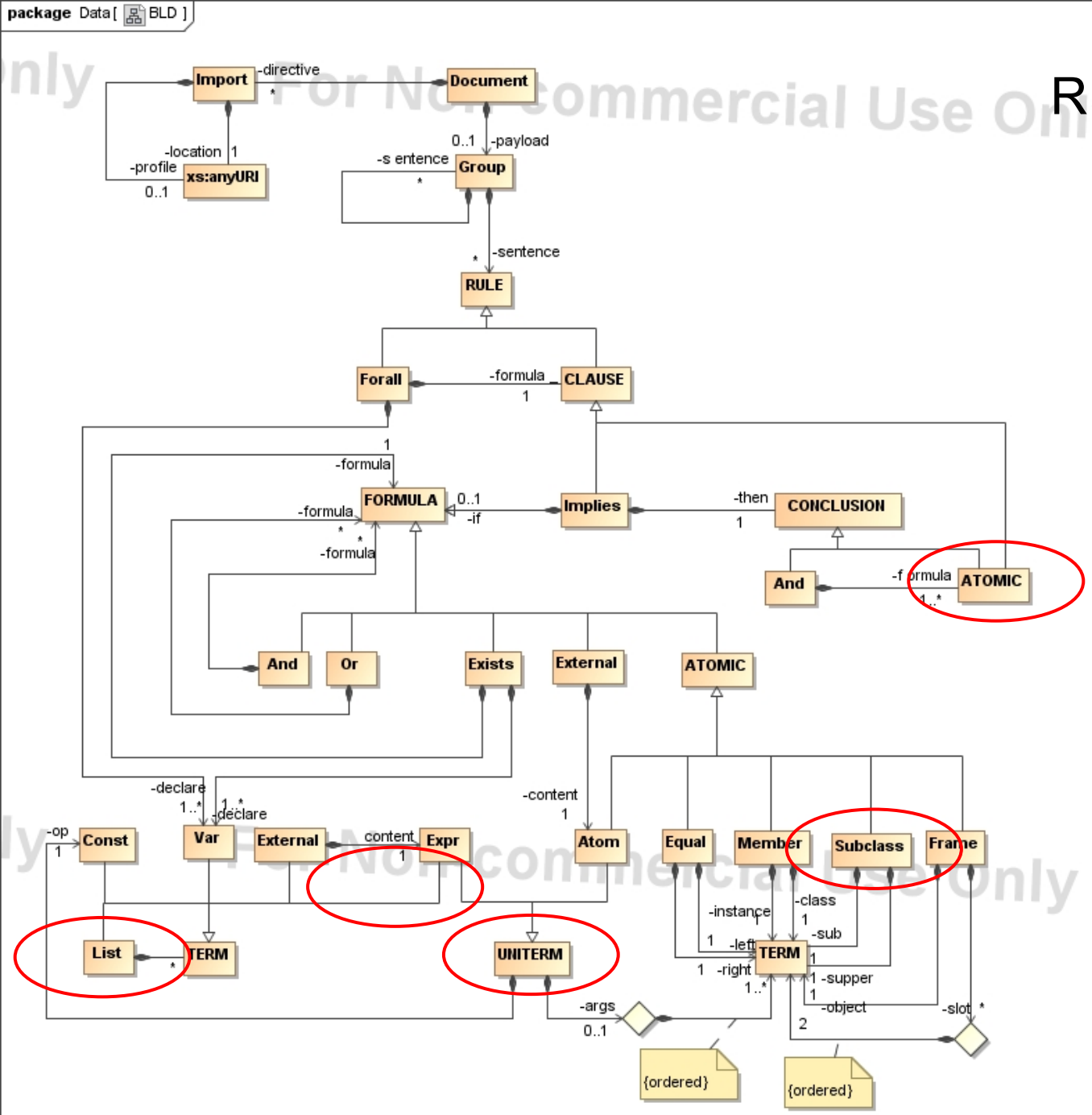
?X["human" -> ?Y]

```
<Frame>
  <object>
    <Var>?X</Var>
  </object>
  <slot ordered="yes">
    <Const type="&xs;string">
      human
    </Const>
    <Var>?Y</Var>
  </slot>
</Frame>
```

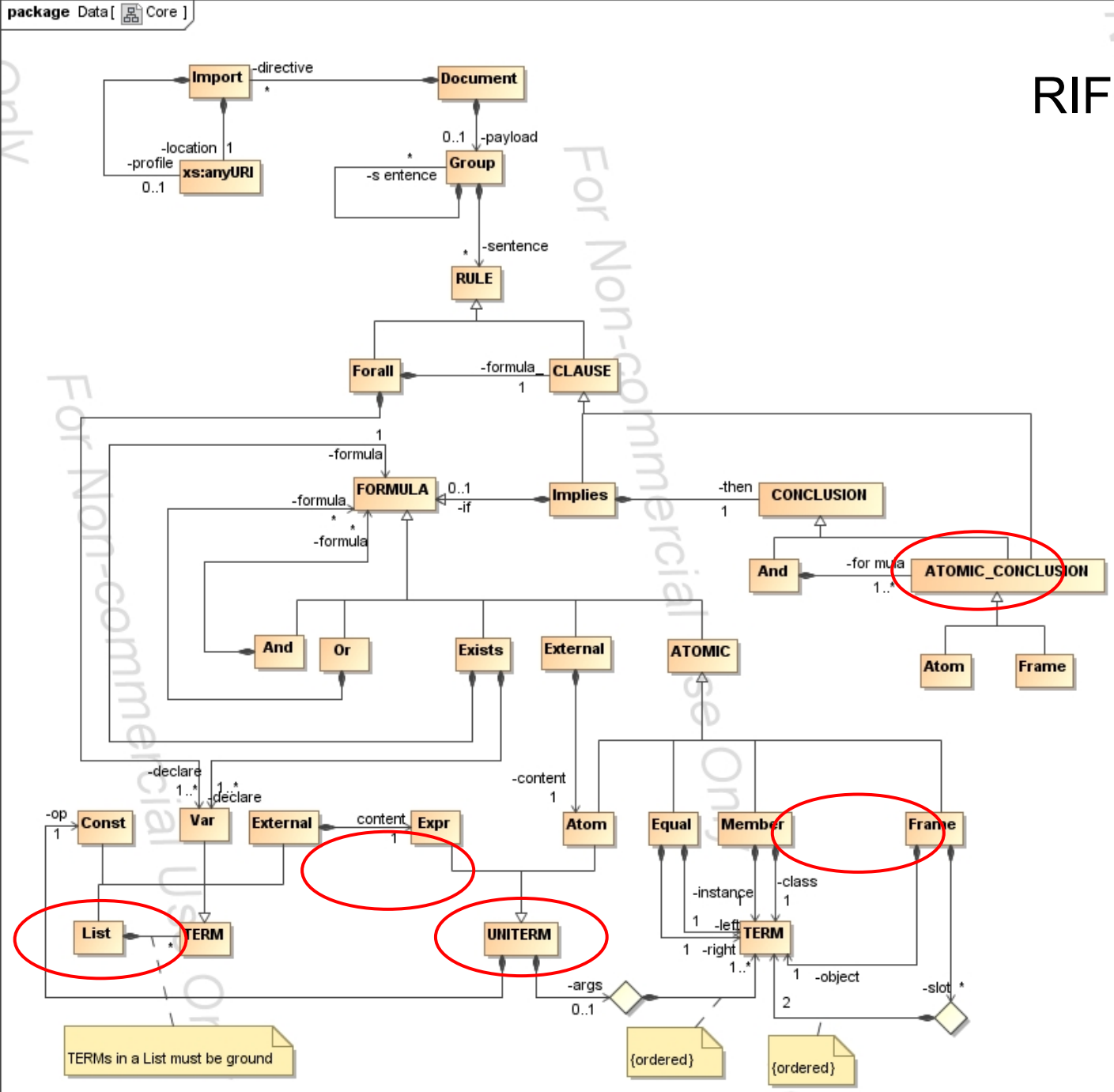
Syntax

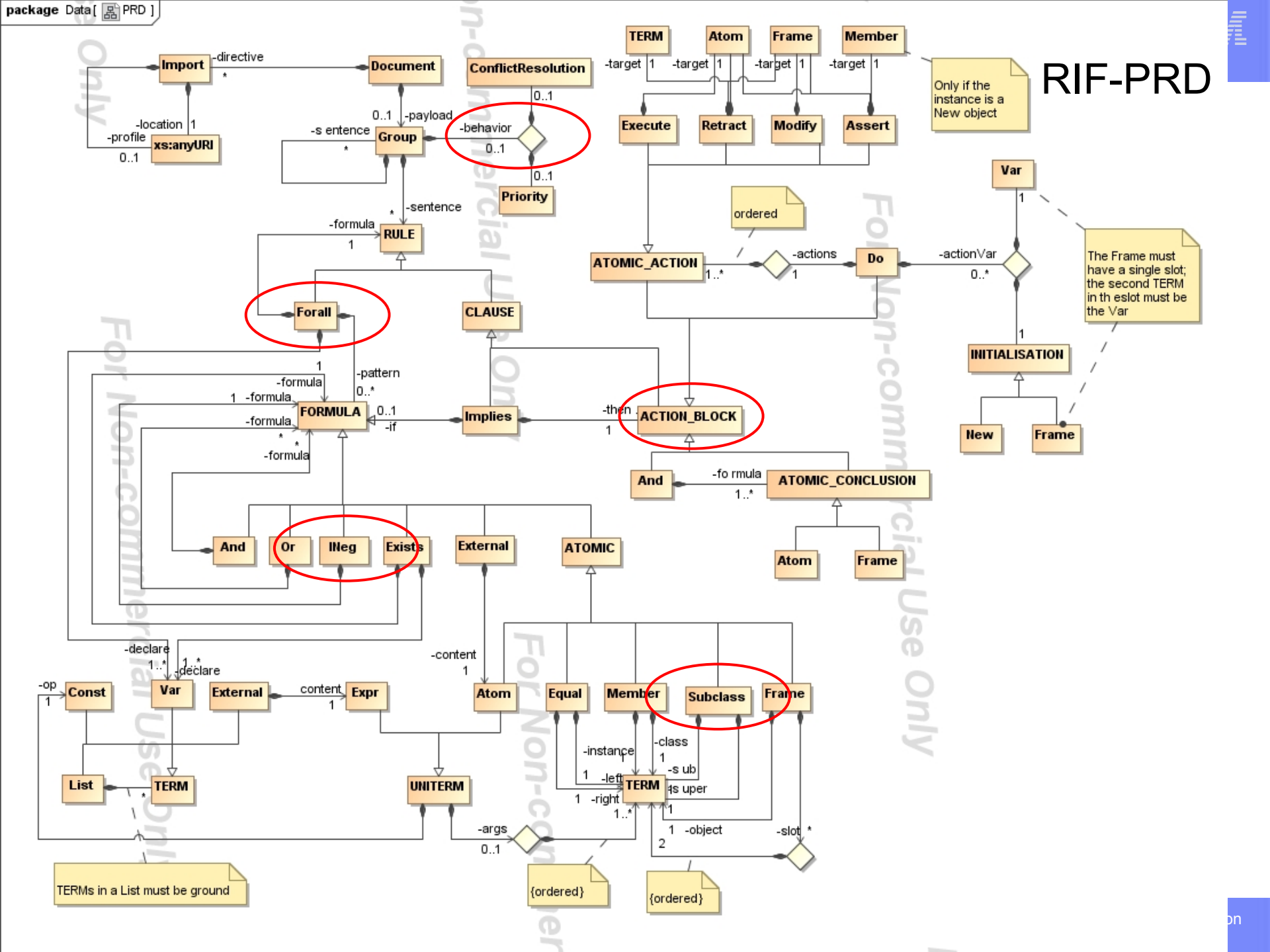
Other dialects

RIF-BLD

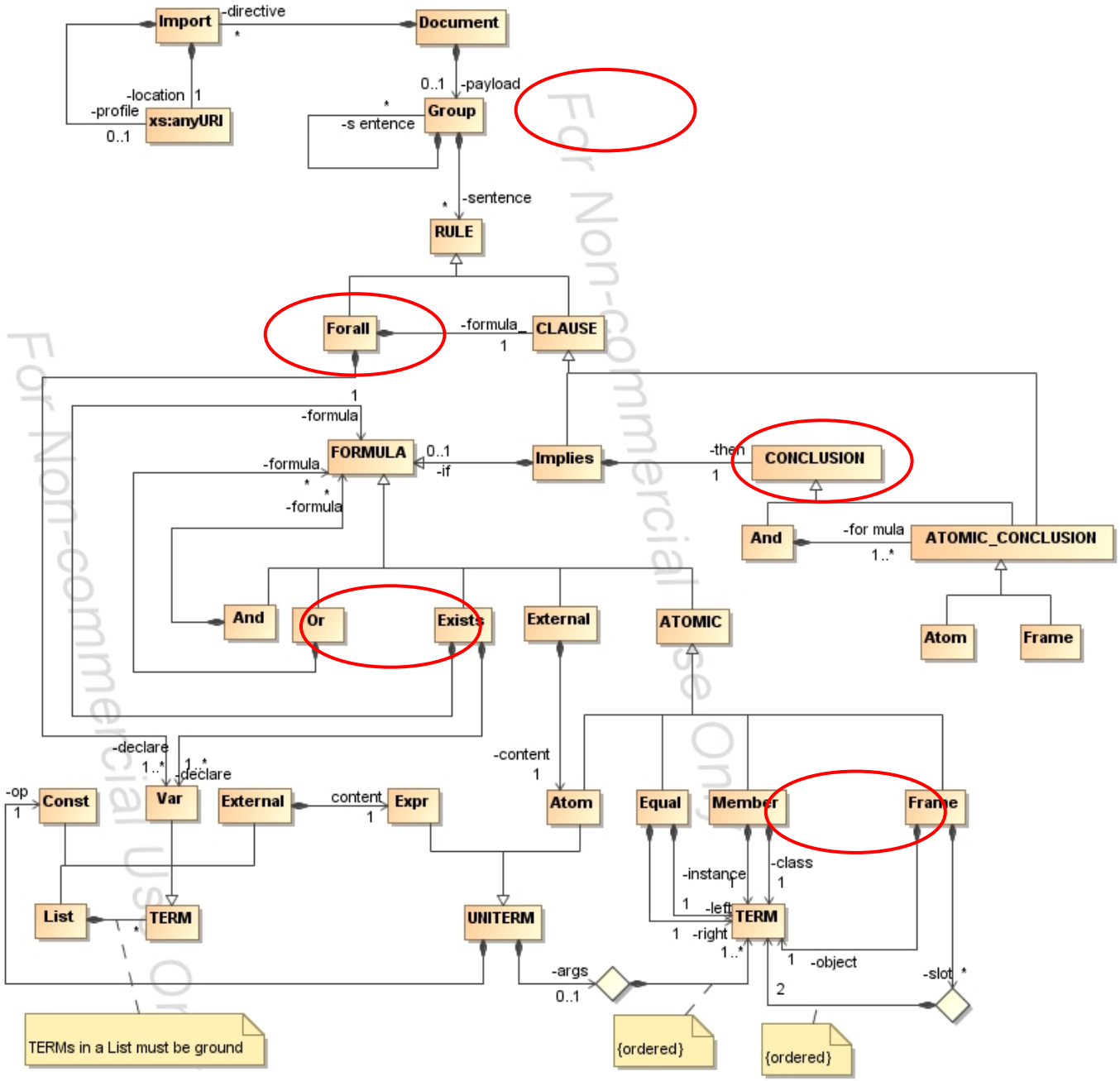

package Data [BLD]
Import
-directive
*
Document
-location
1
-profile
0..1
xs:anyURI
0..1
-payload
-s entence
*
Group
*
-sentence
RULE
Forall
-formula
1
CLAUSE
1
-formula
FORMULA
0..1
-if
Implies
-then
1
CONCLUSION
-formula
*
*
-formula
And
-formula
1..*
ATOMIC
And
Or
Exists
External
ATOMIC
-declare
1..*
1..*
-declare
-content
1
-op
1
Const
Var
External
content
1
Expr
Atom
Equal
Member
Subclass
Frame
List
*
TERM
UNITERM
-instance
1
-left
1
-right
TERM
-class
1
-sub
1
1
-supper
1
-object
2
-slot
*
1..*
-args
0..1
{ordered}
{ordered}

RIF-Core




package Data [PRD]
RIF-PRD
Import
-directive
Document
ConflictResolution
-location
-profile
xs:anyURI
-s entence
Group
-payload
-behavior
Priority
-sentence
-formula
RULE
Forall
CLAUSE
-pattern
FORMULA
Implies
-if
-then
ACTION_BLOCK
And
Or
INeg
Exists
External
ATOMIC
-fo rmula
ATOMIC_CONCLUSION
Atom
Frame
TERM
Atom
Frame
Member
-target
Execute
Retract
Modify
Assert
Only if the instance is a New object
ordered
ATOMIC_ACTION
-actions
Do
-actionVar
Var
The Frame must have a single slot; the second TERM in th eslot must be the Var
INITIALISATION
New
Frame
-declare
-op
Const
Var
External
content
Expr
-content
Atom
Equal
Member
Subclass
Frame
-instance
-class
-s ub
-s uper
-left
-right
TERM
-object
-slot
-args
UNITERM
List
TERM
TERMs in a List must be ground
{ordered}
{ordered}

RIF-Core



Syntax examples

- Look at
  - http://www.w3.org/2005/rules/wiki/Category:Test_Case
- Select test case
- Click on « view XML »

Further extensions, new dialects

- Adding new constructs
  - New elements
  - Additional attributes
  - Namespace?
- Adding new values
  - Additional data types
  - Additional external functions and predicates
  - Additional conflict resolution strategies
  - Additional import profiles
- Must extend RIF Core (at least)
  - Beware of name collision
  - Must extend the semantics consistently
  - For logic dialects, use the RIF framework for logic dialects
  - Forward compatibility?

Decentralized extensibility

- http://lists.w3.org/Archives/Public/www-archive/2009Nov/att-0003

Semantics

Model theoretic semantics of RIF BLD

- Standard first-order semantics with functions and equality
- Interpretations (semantic structures) as usual
  - Domain of discourse
  - Syntax elements onto the domain of discourse mapping
  - Truth valuation function
    - Including =, # and ## axioms (where relevant)
    - Including truth valuation of rule implication and group (as conjunction)
  - Data types and built-in functions and predicates
- Entailment as usual
  - An interpretation I is a model of a formula φ, I |= φ, iff φ is true according to (the truth valuation function in) I
  - A formula φ entails a formula ψ, φ |= ψ, iff every model of φ is a model of ψ

The secret of Michael Jackson death revealed

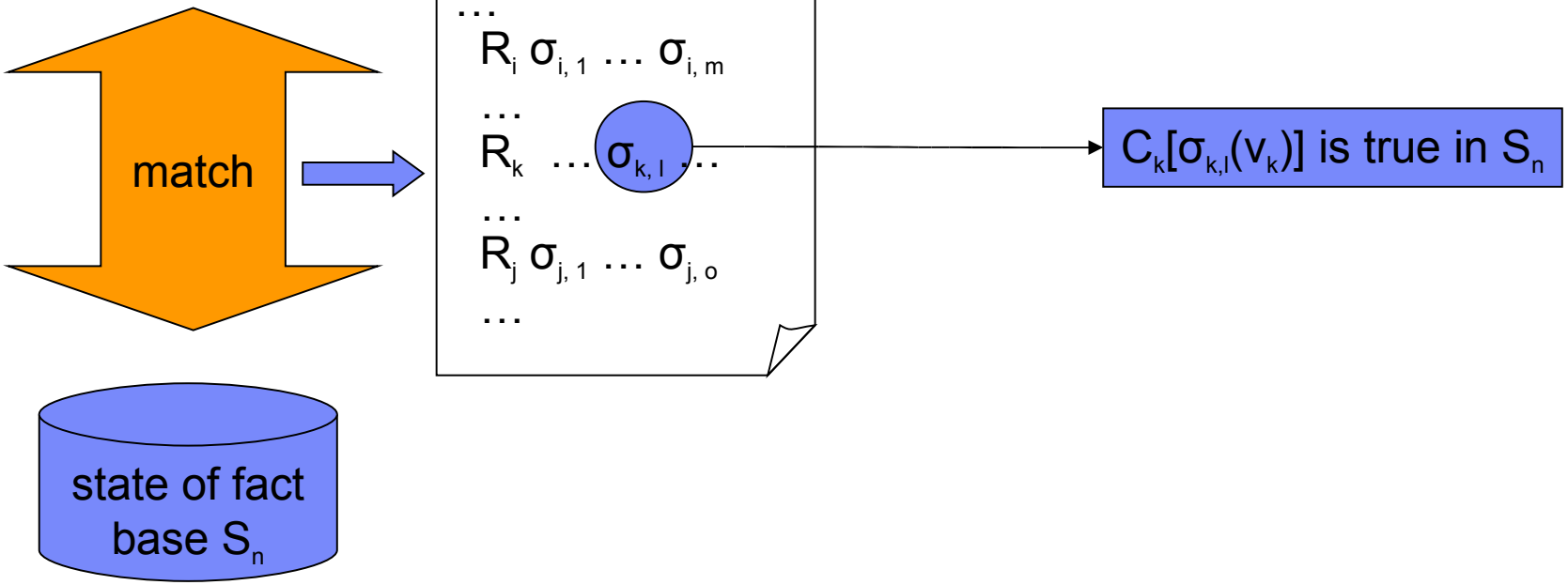
```
Group
  Forall ?Y (?Y[owl:hasProperty -> ex:mortal] :-
             ?Y[owl:hasProperty -> ex:human]

  ex:MichaelJackson[owl:hasProperty -> ex:human]
  ex:Socrates[owl:hasProperty -> ex:human]
```

Entails

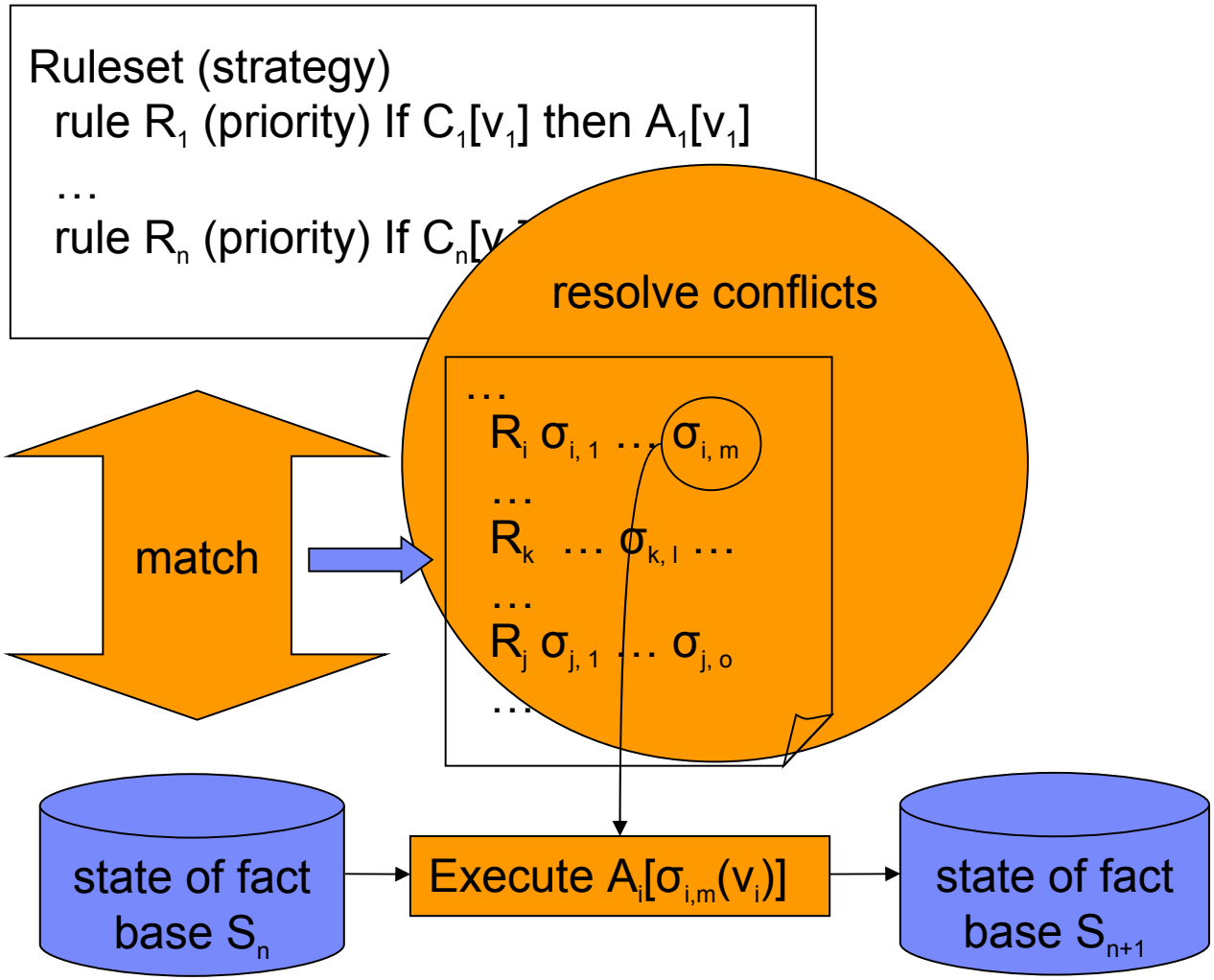
```
ex:MichaelJackson[owl:hasProperty -> ex:human]
ex:MichaelJackson[owl:hasProperty -> ex:mortal]
ex:Socrates[owl:hasProperty -> ex:human]
ex:Socrates[owl:hasProperty -> ex:mortal]
```

Operational semantics of RIF PRD

Ruleset (strategy)
rule R_1 (priority) If $C_1[v_1]$ then $A_1[v_1]$
…
rule R_n (priority) If $C_n[v_n]$then $A_n[v_n]$

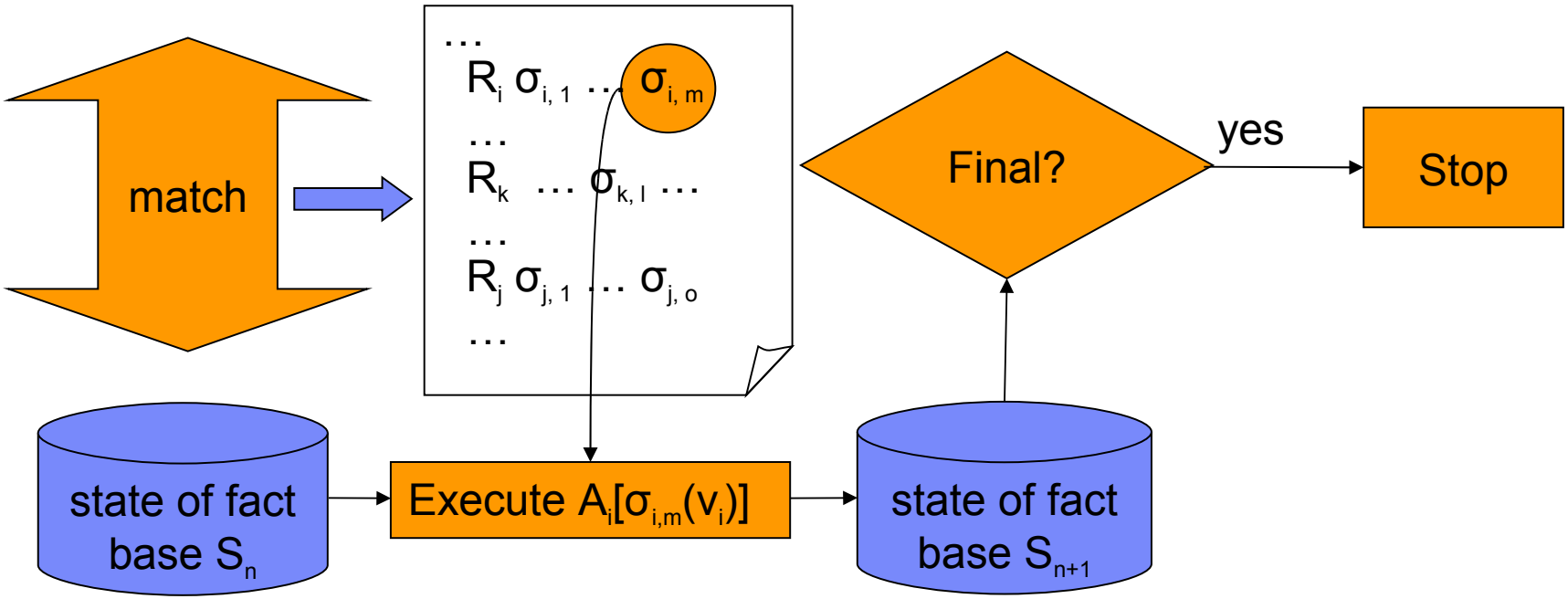


Operational semantics of RIF PRD



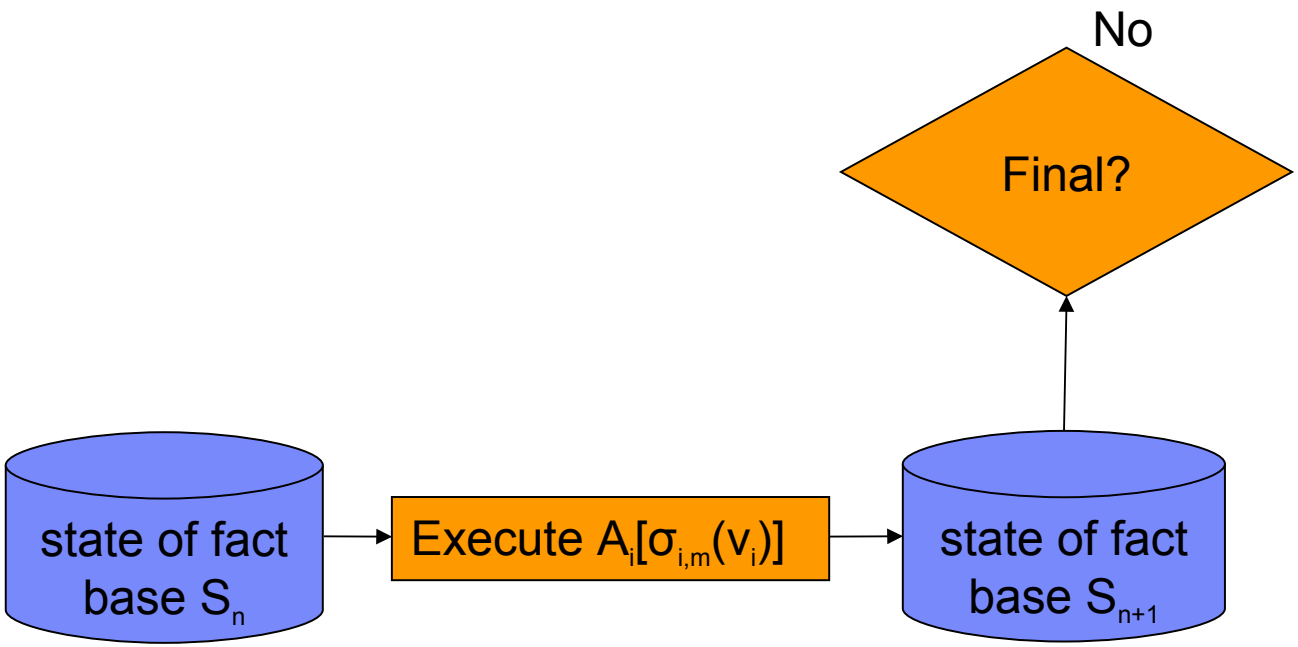
Operational semantics of RIF PRD

Ruleset (strategy)
rule R_1 (priority) If $C_1[v_1]$ then $A_1[v_1]$
…
rule R_n (priority) If $C_n[v_n]$then $A_n[v_n]$

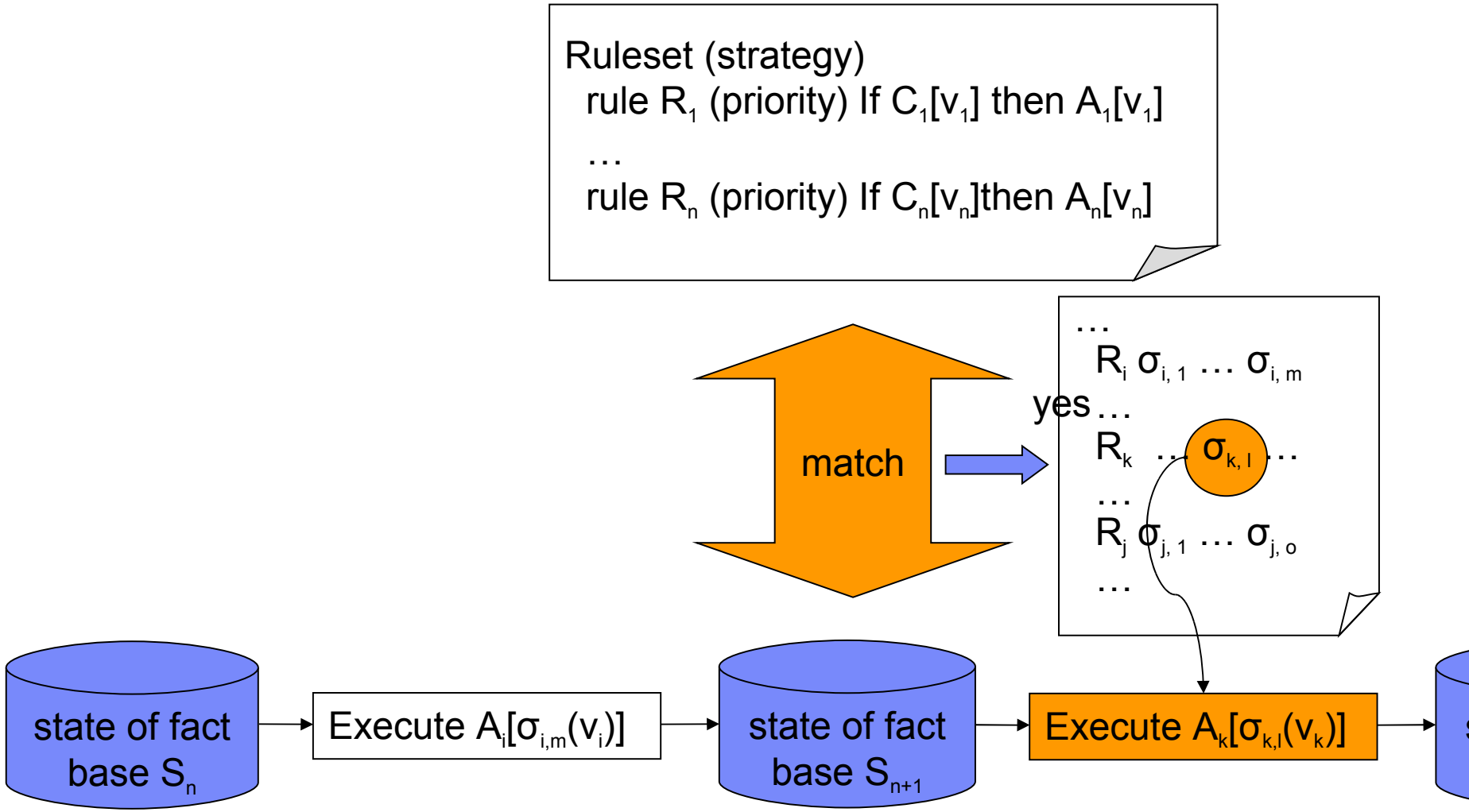


Operational semantics of RIF PRD

Ruleset (strategy)
rule R_1 (priority) If $C_1[v_1]$ then $A_1[v_1]$
…
rule R_n (priority) If $C_n[v_n]$then $A_n[v_n]$



Operational semantics of RIF PRD



The secret of Socrates death revealed

```
Group *R1*
  Forall ?Y (if ?Y[ex:attr -> ex:human]
             then Assert(?Y[ex:attr -> ex:mortal])
            )

S1
  ex:Socrates[ex:attr -> ex:human]
  ex:MichaelJackson[ex:attr -> ex:human]

CS1 = (R1 (?Y/ex:Socrates))
      (R1 (?Y/ex:MichaelJackson))

Execute
  Assert(ex:Socrates[ex:attr -> ex:mortal])
```

The secret of Socrates death revealed

```
Group *R1*
  Forall ?Y (if ?Y[ex:attr -> ex:human]
              then Assert(?Y[ex:attr -> ex:mortal])
             )
```

S2

```
  ex:Socrates[ex:attr -> ex:human]
  ex:Socrates[ex:attr -> ex:mortal]
  ex:MichaelJackson[ex:attr -> ex:human]
```

CS2 = ~~(R1 (?Y/ex:Socrates))~~
(R1 (?Y/ex:MichaelJackson))

Execute

```
  Assert(ex:MichaelJackson[ex:attr -> ex:mortal])
```

The secret of Socrates death revealed

```
Group *R1*
  Forall ?Y (if ?Y[ex:attr -> ex:human]
              then Assert(?Y[ex:attr -> ex:mortal])
             )
```

S3

```
  ex:Socrates[ex:attr -> ex:human]
  ex:Socrates[ex:attr -> ex:mortal]
  ex:MichaelJackson[ex:attr -> ex:human]
  ex:MichaelJackson[ex:attr -> ex:mortal]
```

CS3 = ~~(R1 (?Y/ex:Socrates))~~
~~(R1 (?Y/ex:MichaelJackson))~~

Final

Operational semantics of RIF PRD

- A ***RIF-PRD production rule system*** is defined as a **labeled terminal transition system** *PRS = {S, A, →PRS, T}*, where :
  - *S* is a set of system states;
  - *A* is a set of transition labels, where each transition label is a sequence of ground RIF-PRD atomic actions;
  - The transition relation →PRS ⊆ *S* × *A* × *S*, is defined as follows:
    ∀ (*s*, *a*, *s′*) ∈ *S* × *A* × *S*, (*s*, *a*, *s′*) ∈ →PRS if and only if all of the following hold:
    - *(facts(s), a, facts(s'))* ∈ →*RIF-PRD;
    - *a = actions(picked(s))*;
  - *T* ⊆ *S*, a set of final system states.

Semantics of PRD conditions: “match”

- A *state of the fact base*, w_Φ, is associated to every set of ground atomic formulas, Φ, that satisfies, at least, the # and ## axioms
- A RIF-PRD condition formula ψ is ***satisfied*** in a state of the fact base, *w*, if and only if *w* is represented by a set of ground atomic formulas Φ, and there is a ground substitution σ that matches ψ to Φ
- the ground substitution σ ***matches*** ψ to Φ if and only if one of the following is true:
  - ψ is an atomic formula and either
    - $\sigma(\psi) \in \Phi$, or
    - ψ is an equality formula, t1 = t2, and the ground terms $\sigma(t1)$ and $\sigma(t2)$ have the same value; or
    - ψ is an external atomic formula and the external definition maps $\sigma(\psi)$ to **t** (or *true*),
  - ψ is Not(f) and σ does not match the condition formula f to Φ,
  - ψ is And(f1 ... fn) and either n = 0 or $\forall\, i,\ 1 \le i \le n$, σ matches fi to Φ,
  - ψ is Or(f1 ... fn) and n > *0* and $\exists\, i,\ 1 \le i \le n$, such that σ matches fi to Φ, or
  - ψ is Exists ?v1 ... ?vn (f), and there is a substitution σ' that extends σ in such a way that σ' agrees with σ where σ is defined, and $Var(f) \subseteq Dom(\sigma')$; and σ' matches f to Φ.

Conflict resolution: “pick”

- Find all the instances of the rules that match *w*: conflict set
  - *W* is final if the conflict set is empty
- Select one according to conflict resolution strategy: *picked*

  `rif:forwardChaining`
  - ***Refraction rule**: if ri ∈ cs and lastPicked(ri, s) < recency(ri, s), then cs = cs - ri;*
  - ***Priority rule**: if ri ∈ cs and ri' ∈ cs and priority(ri) < priority(ri'), then cs = cs - ri;*
  - ***Recency rule**: if ri ∈ cs and ri' ∈ cs and recency(ri, s) > recency(ri', s), then cs = cs - ri;*
  - ***Tie-break rule**: if ri ∈ cs, then cs = {ri}.*

Semantics of atomic actions: “execute”

- The semantics of RIF-PRD atomic actions is completely specified by the **transition relation** $\rightarrow$**RIF-PRD** $\subseteq W \times L \times W$, where L is the set of all the ground atomic actions.
- $(w, \alpha, w') \in \rightarrow$*RIF-PRD* if and only if $w \in W$, $w' \in W$, α is a ground atomic action, and one of the following is true:
  - α is **Assert**(φ), where φ is a ground atomic formula, and $w' = w \cup \{\varphi\}$;
  - α is **Retract**(φ), where φ is a ground atomic formula, and $w' = w \setminus \{\varphi\}$;
  - α is **Retract**(o), where o is a constant, and *w′ = w \ {o[s->v] | for all the values of terms s and v} - {o#c | for all the values of term c}*;
  - α is **Modify**(φ), where φ is a ground atomic frame with object o and slot name s, and *w′ = (w \ {o[s->v] | for all the values of term v}) ∪ {φ};*
  - α is **Execute**(φ), where φ is a ground atomic builtin action, and $w' = w$.

Semantics of RIF-Core

- Model-theoretic semantics of RIF-BLD, restricted to RIF-Core syntax
  - Essentially Datalog
  - No subclass, no logic functions, no named argument UNITERMS
  - Only atoms and frames in the head
- Operational semantics of RIF-PRD, restricted to RIF-Core syntax
  - No negation
  - Pattern2Condition equivalence
  - Only Asserts in the RHS: Do2And equivalence
  - With those restriction, the operational semantics of RIF-PRD reduces to Datalog inflationary semantics, equivalent to minimal Herbrand model semantics

Conformance

- Defined in terms of conformant producers, conformant consumers, and conformant documents
- RIF Core and RIF-PRD require **safeness**
  - RIF-BLD does not
  - All variables must be safely bound in the condition before being used in the conclusion

- Unsafe: *Forall X, Y, if X>Y then print X*
- Unsafe: *Forall X, Y, if human(x) then mortal(y)*
- Unsafe: *Forall X, if NOT(mortal(x)) then human(x)*
- Safe: *Forall X, if human(x) AND NOT(dead(x)) then mortal(x)*
- Safe: *Forall X, Y, if human(X) AND X=Y then mortal(Y)*
- Safe: *Forall X, if P(X) AND Exists Y (X>Y) then print X*

Combinations

The Import directive

```
<Import>
    <location> xs:anyURI </location>
    <profile> xs:anyURI </profile>
</Import>
```

- Import RIF documents
- Import RDF and OWL graphs
  - Schema and/or data
- Import XML
  - Schema and/or data

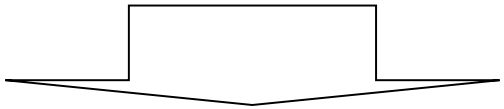
Importing RIF documents

- Semantically equivalent to merging the importing and imported document, with a caveat:
  - `rif:local` constants are local to a document
  - Two `rif:local` constants with the same literal that occur in two different documents are not equal

RIFdoc1.xml

```
<Document xmlns:…>
  <payload>
    <Group>
      <sentence>
        <Atom>
          <op>
            <Const type="&xs;string">
              human
            </Const>
          </op>
          <args ordered="yes">
            <Const type="&rif;local">
              aLocalName
            </Const>
          </args>
        </Atom>
      </sentence>
    </Group>
  </payload>
</Document>
```

```
Document(

  Import(RIFdoc1.)

  Group(

    Forall ?x (
      mortal(?x) :- And( human(?x)
                         ?x = _aLocalName
                       )
                  )
              )
)
```

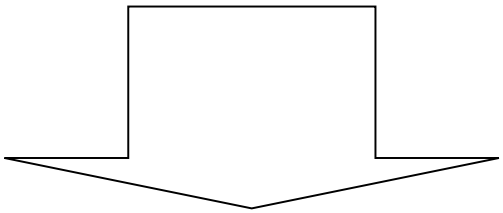


Does not entail mortal(*_aLocalName*)

RIFdoc1.xml

```
<Document xmlns:…>
  <payload>
    <Group>
      <sentence>
        <Atom>
          <op>
            <Const type="&rif;local">
              human
            </Const>
          </op>
          <args ordered="yes">
            <Const type="&xs;string">
              Socrates
            </Const>
          </args>
        </Atom>
      </sentence>
    </Group>
  </payload>
</Document>
```

```
Document(

  Import(RIFdoc1.)

  Group(

    Forall ?x ( mortal(?x) :- _human(?x) )

            )
)
```



Does not entail
mortal(Socrates)

RIF combination with RDF and OWL

- RDF triple *s p o* mapped to frame *s'[p'->o']*
  - *s'[p'->o']* is true iff *s p o* is in the imported RDF graph
  - Condition on data types alignment
  - Simple, RDF, RDFS, D-RDF interpretation iff vocabulary included and axioms satisfied
  - Graph/formula entailed iff satisfied in every interpretation
- OWL 2 Full compatibility is straightforward extension of RDF compatibility
- OWL 2 DL requires syntactic restrictions and semantic extension of RIF frames
  - RIF frame *o[p->v]* is an OWL 2 DL frame iff *p* is a constant and *v* is a constant if *p* is *rdf:type*
  - A variable is DL-safe if it does not occur in a DL frame such that *p* belongs to an imported ontology or *p* is *rdf:type* and *v* belongs to an imported ontology
  - Frame *o[p->v]* is interpreted as relation *p(o, v)* if *p* is not *rdf:type*, and as *o* belonging to set *v* if *p* is *rdf:type*

RIF combination with XML data

- Based on the XPath 2.0 and XQuery 1.0 data model, uses class membership and frames to represent relations between information items in an imported XML data document
  - OID # [<*namespace URI # local name*> | "*local name*"] is true iff OID identifies an element named *local name*, defined in the identified *namespace* or in no namespace
  - OID # <*namespace URI* #type(*local name*)>
  - OID # [<*namespace URI* #type(*local name*)> | "type(*local name*)"] is true iff OID identifies an element of type *local name*, defined in the identified *namespace* or in no namespace
  - OID [*[<namespace URI # local name>* |*"local name"]* -> value] is true iff *value* is the typed content of a sub-element named *local name*, defined in the identified *namespace* or in no namespace, of the element identified by OID, or an identifier of the sub-element itself if the typed value is undefined
  - OID [<*namespace URI* #list(*local name*)> -> value] or
    OID ["list(*local name*)" -> value] is true iff *value* is the ordered list of the typed content of all the sub-elements named *local name*, defined in the identified *namespace* or in no namespace, of the element identified by OID; or of the identifiers of these sub-elements if their typed value is undefined
  - OID [<*namespace URI* #attribute(*local name*)> -> value] or
    OID ["attribute(*local name*)" -> value] is true iff *value* is the typed content of an attribute named *local name*, defined in the identified *namespace* or in no namespace, of the element identified by OID

RIF combination with XML data

http://example.org/customertable/customers.xml

```
<CustomerTable
   xmlns="http://example.org/customertable"
         xmlns:xml=
    "http://www.w3.org/XML/1998/namespace">
  <Customer xml:lang="en">
   <Name> John </Name>
   <Age> 39 </Age>
   <Address>
    <City> London </City>
    <Street> Place Vendôme </Street>
   </Address>
  </Customer>
  <Customer xml:lang="fr">
   <Name> Jane </Name>
   <Age> 22 </Age>
   <Address>
    <City> London </City>
    <Street> Picadilly Circus </Street>
   </Address>
  </Customer>
  <Customer>…</Customer>
  …
</CustomerTable>
```

```
Prefix ex http://example.org/customertable
Prefix xml
   http://www.w3.org/XML/1998/namespace
Prefix rim http://www.w3.org/rif-import

Import ( ex:customers.xml rim:xml-data )

Group

 Forall ?c ?a ?n
  (
   LikelyForeignStudent(?n) :-

   ?c # ex:Customer
   & Exists ?y (?c[ ex:Age -> ?y ] & ?y < 25)
   & ?a # ex:Address
   & ?c [ ex:Address -> ?a ]
   & ?a [ex:City -> "London" ]
   & ?c [ xml:attribute(lang) -> "fr" ]
   & ?c [ ex:Name -> ?n ]
  )
```

RIF combination with XML data

http://example.org/customertable/customers.xml

```
<CustomerTable
   xmlns="http://example.org/customertable"
         xmlns:xml=
     "http://www.w3.org/XML/1998/namespace">
  <Customer xml:lang="en">
   <Name> John </Name>
   <Age> 39 </Age>
   <Address>
    <City> London </City>
    <Street> Place Vendôme </Street>
   </Address>
  </Customer>
  <Customer xml:lang="fr">
   <Name> Jane </Name>
   <Age> 22 </Age>
   <Address>
    <City> London </City>
    <Street> Picadilly Circus </Street>
   </Address>
  </Customer>
  <Customer>…</Customer>
  …
</CustomerTable>
```

```
Prefix ex http://example.org/customertable
Prefix xml
    http://www.w3.org/XML/1998/namespace
Prefix rim http://www.w3.org/rif-import

Import ( ex:customers.xml rim:xml-data )

Group

 Forall ?c ?a ?n
   (
    LikelyForeignStudent(?n) :-

    ?c # ex:Customer
    & Exists ?y (?c[ ex:Age -> ?y ] & ?y < 25)
    & ?a # ex:Address
    & ?c [ ex:Address -> ?a ]
    & ?a [ex:City -> "London" ]
    & ?c [ xml:attribute(lang) -> "fr" ]
    & ?c [ ex:Name -> ?n ]
   )
```

RIF combination with XML data

http://example.org/customertable/customers.xml

```
<CustomerTable
   xmlns="http://example.org/customertable"
         xmlns:xml=
    "http://www.w3.org/XML/1998/namespace">
  <Customer xml:lang="en">
   <Name> John </Name>
   <Age> 39 </Age>
   <Address>
    <City> London </City>
    <Street> Place Vendôme </Street>
   </Address>
  </Customer>
  <Customer xml:lang="fr">
   <Name> Jane </Name>
   <Age> 22 </Age>
   <Address>
    <City> London </City>
    <Street> Picadilly Circus </Street>
   </Address>
  </Customer>
  <Customer>…</Customer>
  …
</CustomerTable>
```

```
Prefix ex http://example.org/customertable
Prefix xml
    http://www.w3.org/XML/1998/namespace
Prefix rim http://www.w3.org/rif-import

Import ( ex:customers.xml rim:xml-data )

Group

 Forall ?c ?a ?n
   (
    LikelyForeignStudent(?n) :-

    ?c # ex:Customer
    & Exists ?y (?c[ ex:Age -> ?y ] & ?y < 25)
    & ?a # ex:Address
    & ?c [ ex:Address -> ?a ]
    & ?a [ex:City -> "London" ]
    & ?c [ xml:attribute(lang) -> "fr" ]
    & ?c [ ex:Name -> ?n ]
   )
```

RIF combination with XML data

http://example.org/customertable/customers.xml

```
<CustomerTable
   xmlns="http://example.org/customertable"
         xmlns:xml=
    "http://www.w3.org/XML/1998/namespace">
  <Customer xml:lang="en">
   <Name> John </Name>
   <Age> 39 </Age>
   <Address>
    <City> London </City>
    <Street> Place Vendôme </Street>
   </Address>
  </Customer>
  <Customer xml:lang="fr">
   <Name> Jane </Name>
   <Age> 22 </Age>
   <Address>
    <City> London </City>
    <Street> Picadilly Circus </Street>
   </Address>
  </Customer>
  <Customer>…</Customer>
  …
</CustomerTable>
```

```
Prefix ex http://example.org/customertable
Prefix xml
    http://www.w3.org/XML/1998/namespace
Prefix rim http://www.w3.org/rif-import

Import ( ex:customers.xml rim:xml-data )

Group

 Forall ?c ?a ?n
   (
    LikelyForeignStudent(?n) :-

    ?c # ex:Customer
    & Exists ?y (?c[ ex:Age -> ?y ] & ?y < 25)
    & ?a # ex:Address
    & ?c [ ex:Address -> ?a ]
    & ?a [ex:City -> "London" ]
    & ?c [ xml:attribute(lang) -> "fr" ]
    & ?c [ ex:Name -> ?n ]
   )
```

RIF combination with XML data

http://example.org/customertable/customers.xml

```
<CustomerTable
   xmlns="http://example.org/customertable"
         xmlns:xml=
    "http://www.w3.org/XML/1998/namespace">
 <Customer xml:lang="en">
  <Name> John </Name>
  <Age> 39 </Age>
  <Address>
   <City> London </City>
   <Street> Place Vendôme </Street>
  </Address>
 </Customer>
 <Customer xml:lang="fr">
  <Name> Jane </Name>
  <Age> 22 </Age>
  <Address>
   <City> London </City>
   <Street> Picadilly Circus </Street>
  </Address>
 </Customer>
 <Customer>…</Customer>
 …
</CustomerTable>
```

```
Prefix ex http://example.org/customertable
Prefix xml
   http://www.w3.org/XML/1998/namespace
Prefix rim http://www.w3.org/rif-import

Import ( ex:customers.xml rim:xml-data )

Group

 Forall ?c ?a ?n
  (
   LikelyForeignStudent(?n) :-

   ?c # ex:Customer
    & Exists ?y (?c[ ex:Age -> ?y ] & ?y < 25)
    & ?a # ex:Address
    & ?c [ ex:Address -> ?a ]
    & ?a [ex:City -> "London" ]
    & ?c [ xml:attribute(lang) -> "fr" ]
    & ?c [ ex:Name -> ?n ]
  )
```

RIF combination with schema valid XML data

http://example.org/customertable/customers.xml

```
<CustomerTable
   xmlns="http://example.org/customertable"
         xmlns:xml=
     "http://www.w3.org/XML/1998/namespace">
  <Customer xml:lang="en">
   <Name> John </Name>
   <Age> 39 </Age>
   <Address>
    <City> London </City>
    <Street> Place Vendôme </Street>
   </Address>
  </Customer>
  <Customer xml:lang="fr">
   <Name> Jane </Name>
   <Age> 22 </Age>
   <Address>
    <City> London </City>
    <Street> Picadilly Circus </Street>
   </Address>
  </Customer>
  <Customer> … </Customer>
  …
</CustomerTable>
```

http://example.org/customertable/customers.xsd

```
<xs:schema xmlns:xs= "…/XMLSchema"
        xmlns:xml="…/XML/1998/namespace"

        targetNamespace=
           "http://example.org/customertable">
  <xs:element name="Name" type=…/>
  <xs:element name="Age" type="xs:int"/>
  <xs:element name="Address">
    …
  </xs:element>
  <xs:element name="Customer">
   <xs:complexType>
    <xs:sequence>
     <xs:element ref="Name"/>
     <xs:element ref="Age"/>
     <xs:element ref="Address"/>
    </xs:sequence>
   </xs:complexType>
   <xs:attribute ref="xml:lang"/>
  </xs:element>
  <xs:element name="CustomerTable">
    …
  </xs:element>
</xs:schema>
```

RIF combination with schema valid XML data

```
Prefix ex http://example.org/customertable
Prefix xml
    http://www.w3.org/XML/1998/namespace
Prefix rim http://www.w3.org/rif-import

Import ( ex:customers.xml
         rim:xml-schema-valid-data
         ex:customers.xsd )

Group

 Forall ?c ?a ?n
   (
     LikelyForeignStudent(?n) :-

     ?c # ex:Customer
     & Exists ?y (?c[ ex:Age -> ?y ] & ?y < 25)
     & ?a # ex:Address
     & ?c [ ex:Address -> ?a ]
     & ?a [ex:City -> "London" ]
     & ?c [ xml:attribute(lang) -> "fr" ]
     & ?c [ ex:Name -> ?n ]
   )
```

http://example.org/customertable/customers.xsd

```
<xs:schema xmlns:xs= "…/XMLSchema"
        xmlns:xml="…/XML/1998/namespace"

        targetNamespace=
            "http://example.org/customertable">
  <xs:element name="Name" type=…/>
  <xs:element name="Age" type="xs:int"/>
  <xs:element name="Address">
    …
  </xs:element>
  <xs:element name="Customer">
   <xs:complexType>
    <xs:sequence>
     <xs:element ref="Name"/>
     <xs:element ref="Age"/>
     <xs:element ref="Address"/>
    </xs:sequence>
   </xs:complexType>
   <xs:attribute ref="xml:lang"/>
  </xs:element>
  <xs:element name="CustomerTable">
    …
  </xs:element>
</xs:schema>
```

RIF combination with schema valid XML data

http://example.org/customertable/customers.xml

```
<CustomerTable
   xmlns="http://example.org/customertable"
         xmlns:xml=
     "http://www.w3.org/XML/1998/namespace">
  <Customer xml:lang="en">
   <Name> John </Name>
   <Age> 39 </Age>
   <Address>
    <City> London </City>
    <Street> Place Vendôme </Street>
   </Address>
  </Customer>
  <Customer xml:lang="fr">
   <Name> Jane </Name>
   <Age> 22 </Age>
   <Address>
    <City> London </City>
    <Street> Picadilly Circus </Street>
   </Address>
  </Customer>
  <Customer>…</Customer>
  …
</CustomerTable>
```

```
<xs:schema xmlns:xs= "…/XMLSchema"
        xmlns:xml="…/XML/1998/namespace"

        targetNamespace=
            "http://example.org/customertable">
   <xs:element name="Name" type=…/>
   <xs:element name="Age" type="xs:string"/>
   <xs:element name="Address">
```

```
Forall ?c ?a ?n
  (
   LikelyForeignStudent(?n) :-

   ?c # ex:Customer
   & Exists ?y (?c[ ex:Age -> ?y ] & ?y < 25)
   & ?a # ex:Address
   & ?c [ ex:Address -> ?a ]
   & ?a [ex:City -> "London" ]
   & ?c [ xml:attribute(lang) -> "fr" ]
   & ?c [ ex:Name -> ?n ]
  )
```

undefined!

RIF combination with XML schema

Prefix ex http://example.org/customertable
Prefix xml
http://www.w3.org/XML/1998/namespace
Prefix rim http://www.w3.org/rif-import

Import (-
rim:xml-schema
ex:customers.xsd)

Group

Forall ?c ?a ?n
(
LikelyForeignStudent(?n) :-

?c # ex:Customer
& Exists ?y (?c[ex:Age -> ?y] & ?y < 25)
& ?a # ex:Address
& ?c [ex:Address -> ?a]
& ?a [ex:City -> "London"]
& ?c [xml:attribute(lang) -> "fr"]
& ?c [ex:Name -> ?n]
)

http://example.org/customertable/customers.xsd

```
<xs:schema xmlns:xs= "…/XMLSchema"
        xmlns:xml="…/XML/1998/namespace"

        targetNamespace=
            "http://example.org/customertable">
  <xs:element name="Name" type=…/>
  <xs:element name="Age" type="xs:int"/>
  <xs:element name="Address">
    …
  </xs:element>
  <xs:element name="Customer">
   <xs:complexType>
    <xs:sequence>
     <xs:element ref="Name"/>
     <xs:element ref="Age"/>
     <xs:element ref="Address"/>
    </xs:sequence>
   </xs:complexType>
   <xs:attribute ref="xml:lang"/>
  </xs:element>
  <xs:element name="CustomerTable">
    …
  </xs:element>
</xs:schema>
```

RIF combination any data, based on XML schema

Prefix ex http://example.org/customertable
Prefix xml
http://www.w3.org/XML/1998/namespace
Prefix rim http://www.w3.org/rif-import

Import (ex:customers.data
rim:xml-schema
ex:customers.xsd)

Group

```
Forall ?c ?a ?n
  (
    LikelyForeignStudent(?n) :-

    ?c # ex:Customer
    & Exists ?y (?c[ ex:Age -> ?y ] & ?y < 25)
    & ?a # ex:Address
    & ?c [ ex:Address -> ?a ]
    & ?a [ex:City -> "London" ]
    & ?c [ xml:attribute(lang) -> "fr" ]
    & ?c [ ex:Name -> ?n ]
  )
```

http://example.org/customertable/customers.xsd

```
<xs:schema xmlns:xs= "…/XMLSchema"
        xmlns:xml="…/XML/1998/namespace"

        targetNamespace=
            "http://example.org/customertable">
   <xs:element name="Name" type=…/>
   <xs:element name="Age" type="xs:int"/>
   <xs:element name="Address">


                                    'Customer">
```

http://example.org/customertable/customers.data

CustId	Name	Age	AddrId
C0001	John	39	A0001
C0002	Jane	22	A0002

AddrId	City	Street
A0001	Paris	Place Vendôme
A0002	London	Picadilly Circus

Conclusion

CR RIF dialects are basic, but foundational

Issues

- Documenting extensions
  - Importing externals?
  - Forward compatibility?
- Combinatorics of styles
  - Esp. for non-semantics, e.g. rule names
  - Profiles?

Next steps

- Resolving issues
- PRD
  - Object orientation, aggregates (collect), events…
  - Alignment with OMG PRR
- BLD
  - Negation, FOL…
- Combination with UML data models
- XTAN (XML transform as needed)
- …

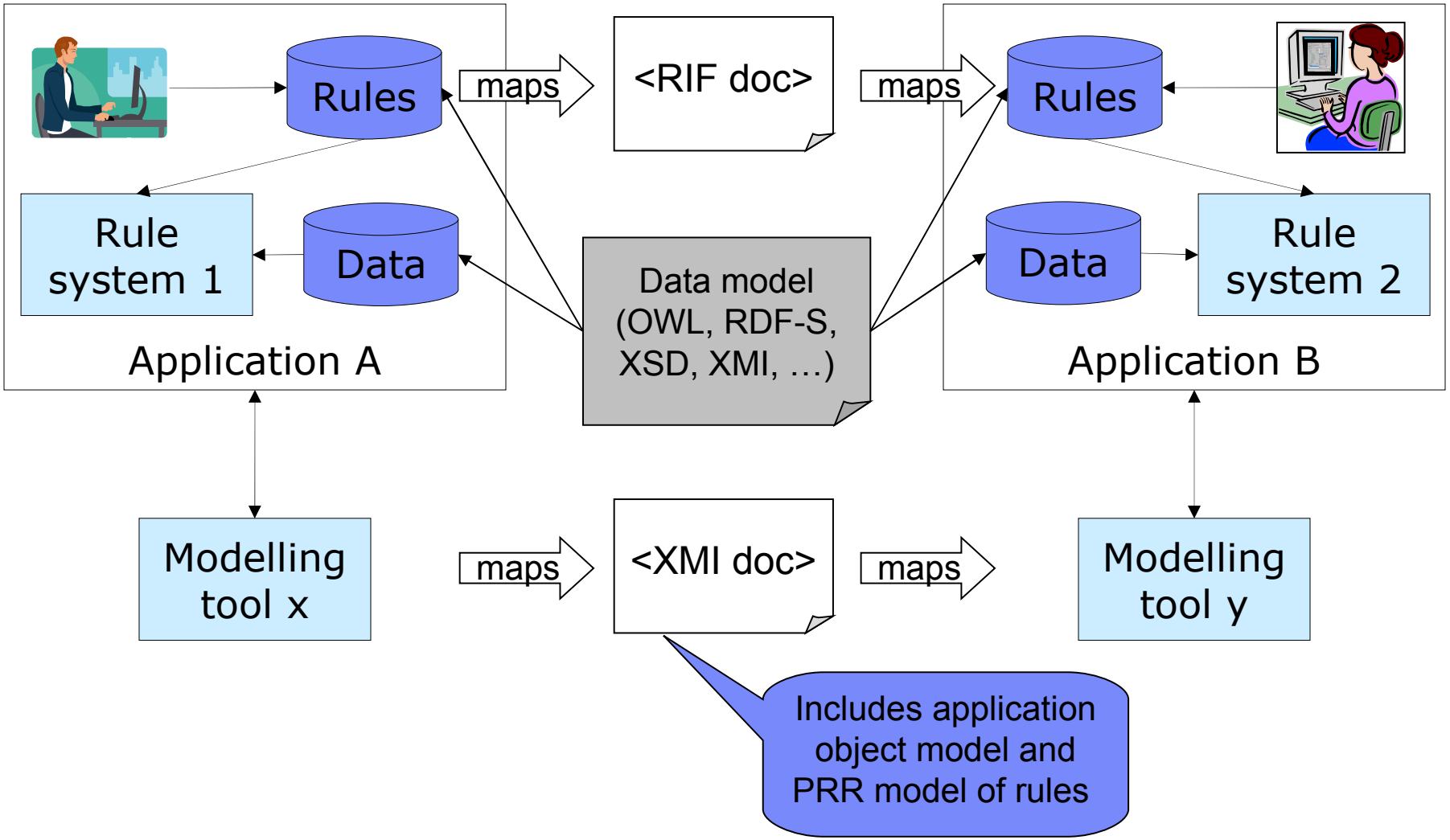
- **Implement!**
  - Report on implementations
  - http://www.w3.org/2005/rules/wiki/Implementations

Thank you!

Questions?

Back-up slides

RIF vs PRR



Differences: RIF vs PRR

- PRR: a rule is (just another) class in a model
  - PRR provides a standard means for including rules into the model of an application (at design time)
  - PRR OCL to constrain allowed instances (objects) at runtime
- RIF: a rule is (just another) data item
  - RIF provides a standard means to feed rules into an application (at run time)
  - Semantics to prescribe (intended) application's behaviour
- PRR focused on 1 rule type based on existing use
  - Generic model defined for extensibility
- RIF intended to cover many rule types and uses
  - Dialects defined for implementability