

# Object-Oriented RuleML

Re-Modularized and XML Schematized via Content Models

David Hirtle  
Coop student, NRC IIT e-Business

December 2, 2003



National Research  
Council Canada

Conseil national  
de recherches Canada

Canada

# Overview

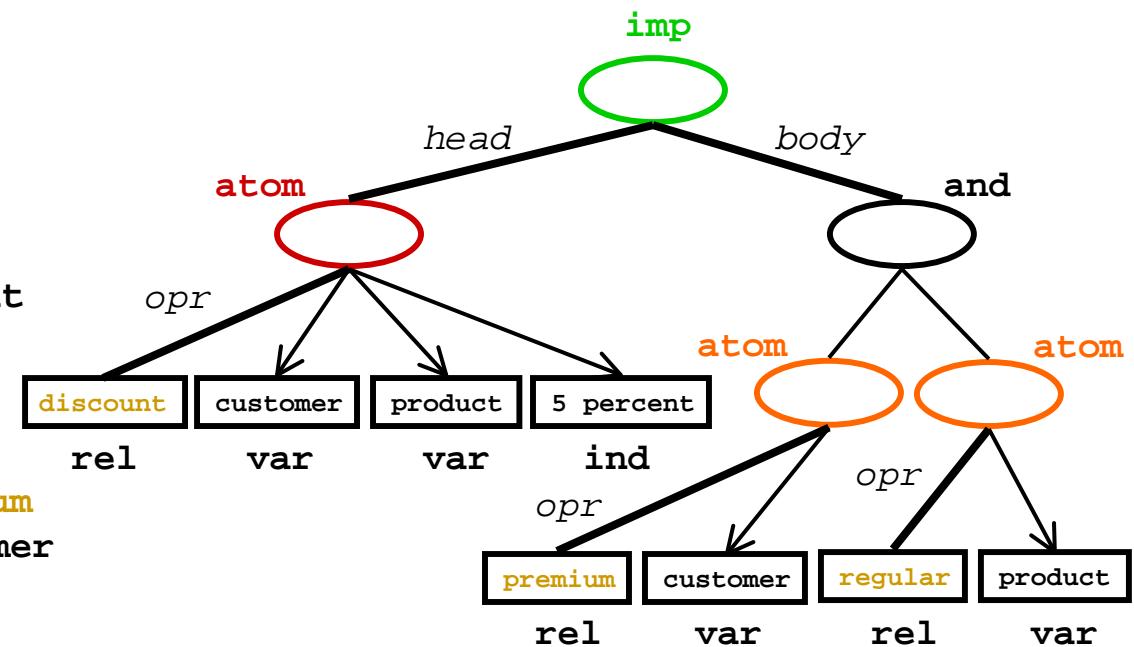
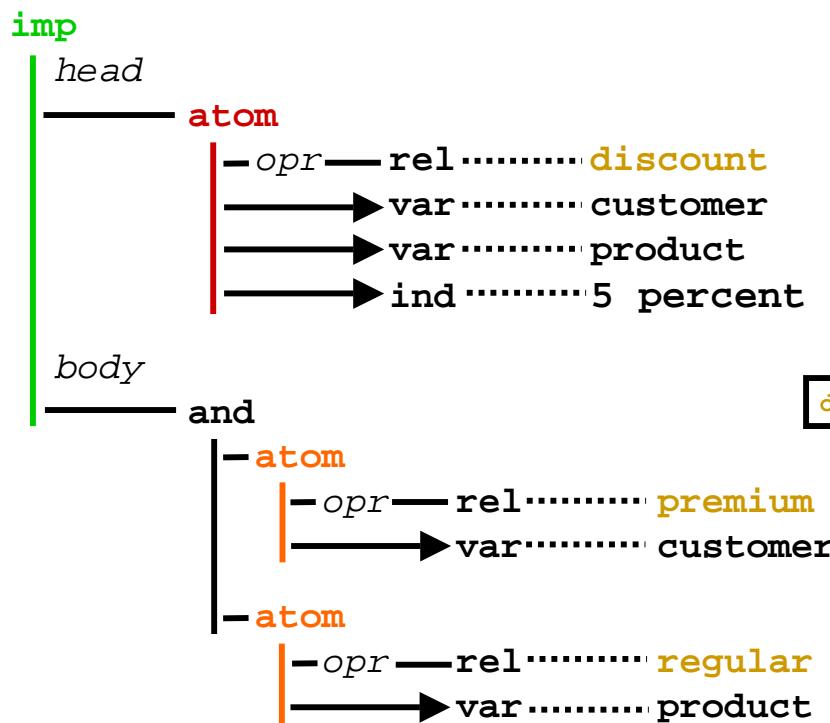
- RuleML → (W)OO RuleML
- DTDs
  - (W)OO extension
  - remodularization
  - inheritance
  - content models
  - demo
- XML Schema
  - inheritance
  - content models
  - demo
- Steering Committee
- Future Work

## RuleML - Quick summary

- rules are essential for the Semantic Web
  - inference rules
  - transformation rules
- rule interchange is important for e-Business
- Rule Markup Initiative aims to define a canonical language (RuleML) for interoperable rule markup
  - XSLT translators to other SW languages
- collaborating with W3C and other standards bodies
- more information: [[www.ruleml.org](http://www.ruleml.org)]

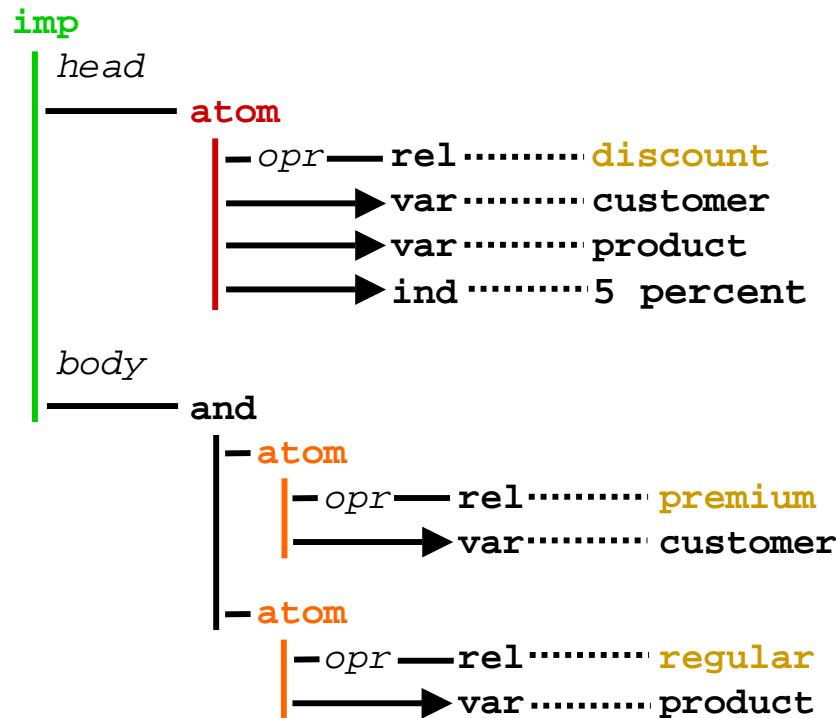
## RuleML - Example

"The **discount** for a *customer* buying a *product* is **5 percent** if the *customer* is **premium** and the *product* is **regular**."



## RuleML - Example

"The **discount** for a *customer* buying a *product* is **5 percent** if the *customer* is **premium** and the *product* is **regular**."



```

<imp>
  <_head>
    <atom>
      <_opr><rel>discount</rel></_opr>
      <var>customer</var>
      <var>product</var>
      <ind>5.0 percent</ind>
    </atom>
  </_head>
  <_body>
    <and>
      <atom>
        <_opr><rel>premium</rel></_opr>
        <var>customer</var>
      </atom>
      <atom>
        <_opr><rel>regular</rel></_opr>
        <var>product</var>
      </atom>
    </and>
  </_body>
</imp>

```

# OO RuleML

- Object-Oriented extension to RuleML
  - non-positional user-level roles (metarole `_r`)

```
<atom>
  <_opr><rel>discount</rel></_opr>
  <_r n="amount"><ind>5.0 percent</ind></r>
  <_r n="product name"><var>product</var></r>
  <_r n="customer name"><var>customer</var></r>
</atom>
```

- term typing

```
<atom>
  <_opr><rel>discount</rel></_opr>
  <_r n="customer name"><var type="Cust">customer</var></r>
  <_r n="product name"><var type="Prod">product</var></r>
  <_r n="amount"><ind type="Fixed_Percent">5.0 percent</ind></r>
</atom>
```

- URI-grounding

```
...
<_opr><rel href="example.com/discounts">discount</rel></_opr>
...
```

[[www.cs.unb.ca/~boley/ruleml/ruleml-rgs.pdf](http://www.cs.unb.ca/~boley/ruleml/ruleml-rgs.pdf)]

# WOO RuleML

- Weighted extension to Object-Oriented RuleML

```
<atom>
  <_opr><rel href="example.com/discounts">discount</rel></_opr>
  <_r n="customer name" w="0.2"><var type="Cust">customer</var></r>
  <_r n="product name" w="0.2"><var type="Prod">product</var></r>
  <_r n="amount" w="0.6"><ind type="Fixed_Percent">5.0 percent</ind></r>
</atom>
```

# Document Type Definition (DTD)

- XML is based on user-defined elements
  - anything goes?
- DTDs define structure/schema/grammar
  - in other words, which elements are allowed where
- “well-formed” vs. “valid”
  - well-formed XML just follows proper syntax
  - **valid** XML is well-formed and conforms to DTD
- need DTD(s) to define structure of RuleML

# DTDs - Meta-Syntax

- similar to Extended Backus-Naur Form (EBNF)
  - basic meta-syntax is: `<!ELEMENT name (content)>`  
e.g. a var(iable) consists of any old string

- more meta-syntax: , | \* + ? zero or one  
e.g. an atom consists of an opr followed by zero or more inds or vars

```
<!ELEMENT atom (_opr, (ind 0 var)*>)
          choice
          sequence      zero or more
```

- attributes: `<!ATTLIST elem_name attr_name type use>`  
e.g. `<!ATTLIST ind href CDATA #IMPLIED>`

# DTDs - WOO RuleML Changes

- user-level roles

```
<!ELEMENT atom (  
    ( _opr,  
      (_r)*, ( (ind | var | cterm | tup)+, (_r)* )?  
    )  
  |  
  ( ( ( _opr,  
        (( _r )+, ( (ind | var | cterm | tup)+, (_r)* )?)  
      |  
      ((ind | var | cterm | tup)+, (_r)* )  
    ),  
    _opr  
  )  
)>
```

# DTDs - WOO RuleML Changes

- user-level roles

```
<!ELEMENT atom ... as before ... >
<!ELEMENT cterm (
    ( _opc,
      (_r)*, ( (ind | var | cterm | tup)+, (_r)* )?
    )
  |
  (
    (
      ((_r)+, ( (ind | var | cterm | tup)+, (_r)* )?)
    |
      ((ind | var | cterm | tup)+, (_r)*)
    ),
    _opc
  )
)>
```

## DTDs - WOO RuleML Changes

- user-level roles

```
<!ELEMENT atom ... as before ... >
<!ELEMENT cterm ... as before ... >
<!ELEMENT tup (
    (_r)*, ( (ind | var | cterm | tup)+, (_r)* )?
)>
```

## DTDs - WOO RuleML Changes

- user-level roles

```
<!ELEMENT atom ... as before ... >
<!ELEMENT cterm ... as before ... >
<!ELEMENT tup (
    (_r)*, ( (ind | var | cterm | tup)+, (_r)* )?
)>

<!ELEMENT _r (ind | var | cterm | tup)>
<!ATTLIST _r n CDATA #REQUIRED>
<!ATTLIST _r card CDATA #IMPLIED>
```

## DTDs - WOO RuleML Changes

- user-level roles

```
<!ELEMENT atom ... as before ... >
<!ELEMENT cterm ... as before ... >
<!ELEMENT tup (
    (_r)*, ( (ind | var | cterm | tup)+, (_r)* )?
)>

<!ELEMENT _r (ind | var | cterm | tup)>
<!ATTLIST _r n CDATA #REQUIRED>
<!ATTLIST _r card CDATA #IMPLIED>
```

- term typing

```
<!ATTLIST ind type CDATA #IMPLIED>
<!ATTLIST var type CDATA #IMPLIED>
<!ATTLIST cterm type CDATA #IMPLIED>
```

# DTDs - WOO RuleML Changes

- user-level roles

```
<!ELEMENT atom ... as before ... >
<!ELEMENT cterm ... as before ... >
<!ELEMENT tup (
    (_r)*, ( (ind | var | cterm | tup)+, (_r)* )?
)>

<!ELEMENT _r (ind | var | cterm | tup)>
<!ATTLIST _r n CDATA #REQUIRED>
<!ATTLIST _r card CDATA #IMPLIED>
```

- term typing

```
<!ATTLIST ind type CDATA #IMPLIED>
<!ATTLIST var type CDATA #IMPLIED>
<!ATTLIST cterm type CDATA #IMPLIED>
```

- URI-grounding: <!ATTLIST ind href CDATA #IMPLIED>  
  <!ATTLIST rel href CDATA #IMPLIED>  
  <!ATTLIST ctor href CDATA #IMPLIED>

# DTDs - WOO RuleML Changes

- user-level roles

```
<!ELEMENT atom ... as before ... >
<!ELEMENT cterm ... as before ... >
<!ELEMENT tup (
                  (_r)*, ( (ind | var | cterm | tup)+, (_r)* )?
              )>

<!ELEMENT _r (ind | var | cterm | tup)>
<!ATTLIST _r n CDATA #REQUIRED>
<!ATTLIST _r card CDATA #IMPLIED>
```

- term typing

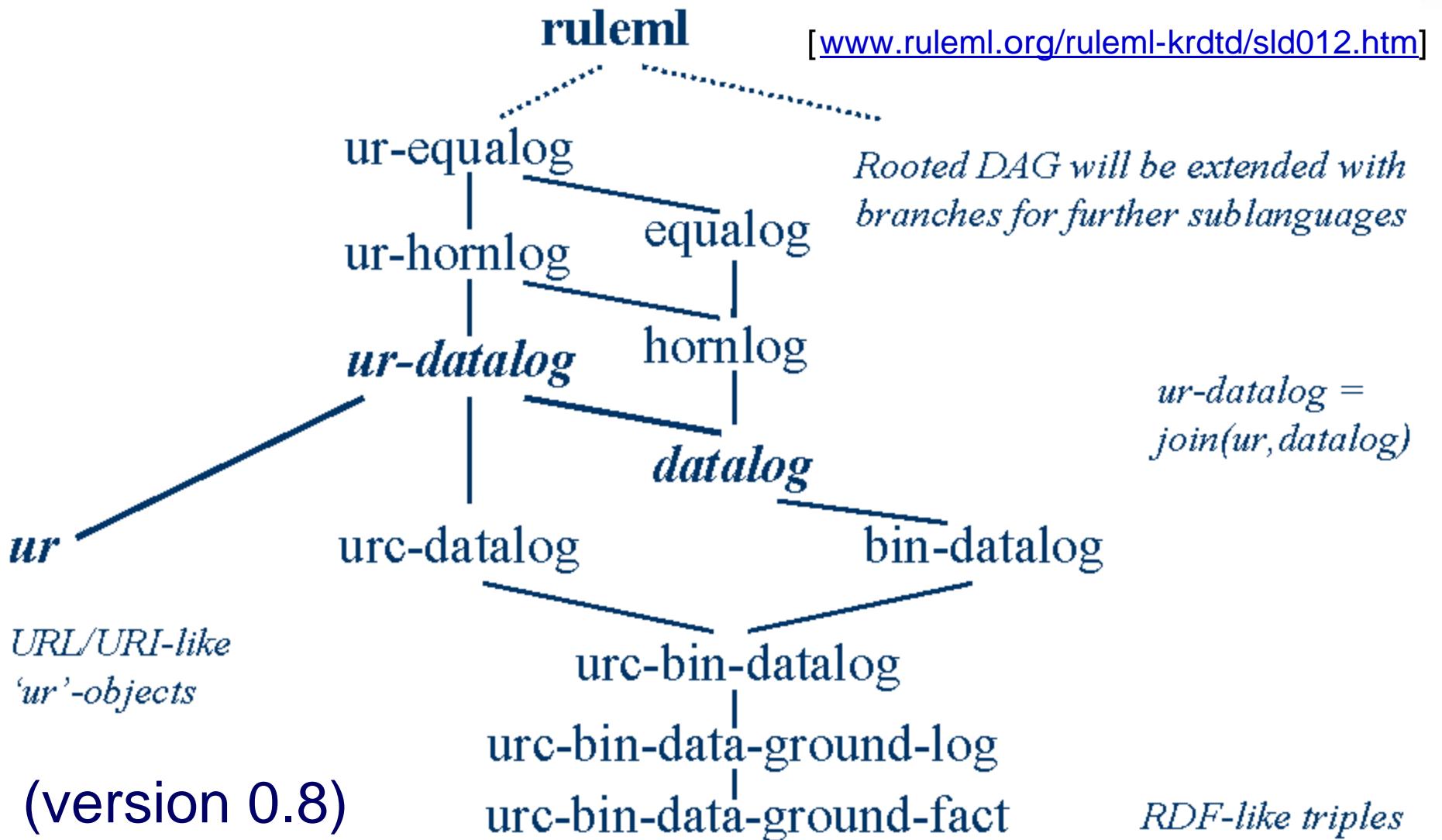
```
<!ATTLIST ind type CDATA #IMPLIED>
<!ATTLIST var type CDATA #IMPLIED>
<!ATTLIST cterm type CDATA #IMPLIED>
```

- URI-grounding:  
    <!ATTLIST **ind** href CDATA #IMPLIED>  
    <!ATTLIST **rel** href CDATA #IMPLIED>  
    <!ATTLIST **ctor** href CDATA #IMPLIED>

- weighted extension: `<!ATTLIST _r w CDATA #IMPLIED>`

## DTDs - Modularization

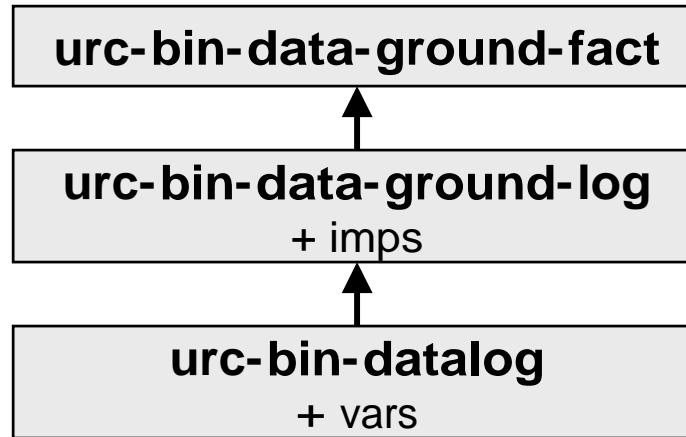
- a family of DTD modules instead of a single large DTD
- modularization has advantages
  - accommodate rule subcommunities
  - each node in hierarchy represents well-known rule system (datalog, hornlog, equalog ...)
  - specificity, increase interoperability



## DTDs - Modularization

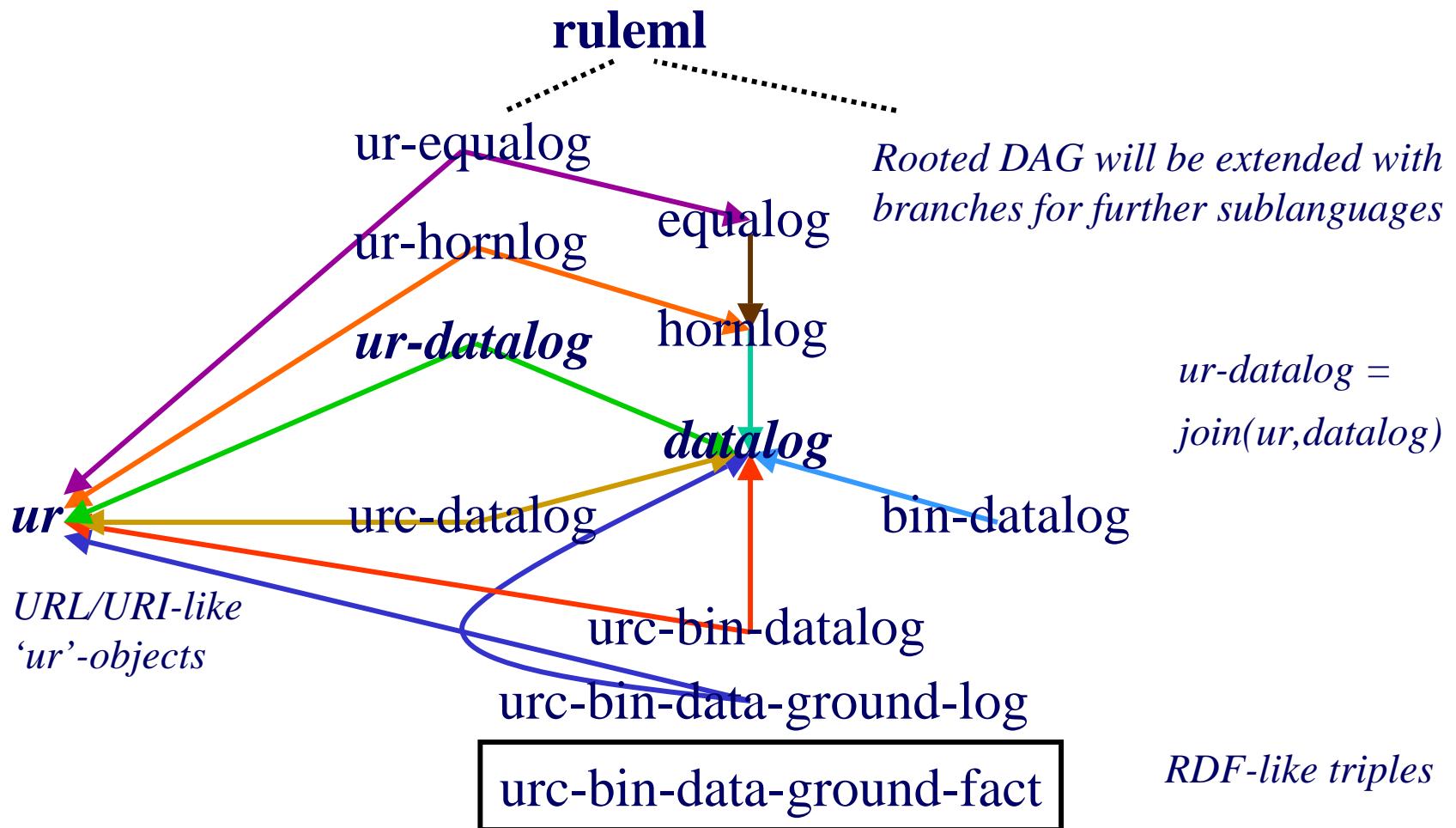
- modules inherit from one another

e.g.



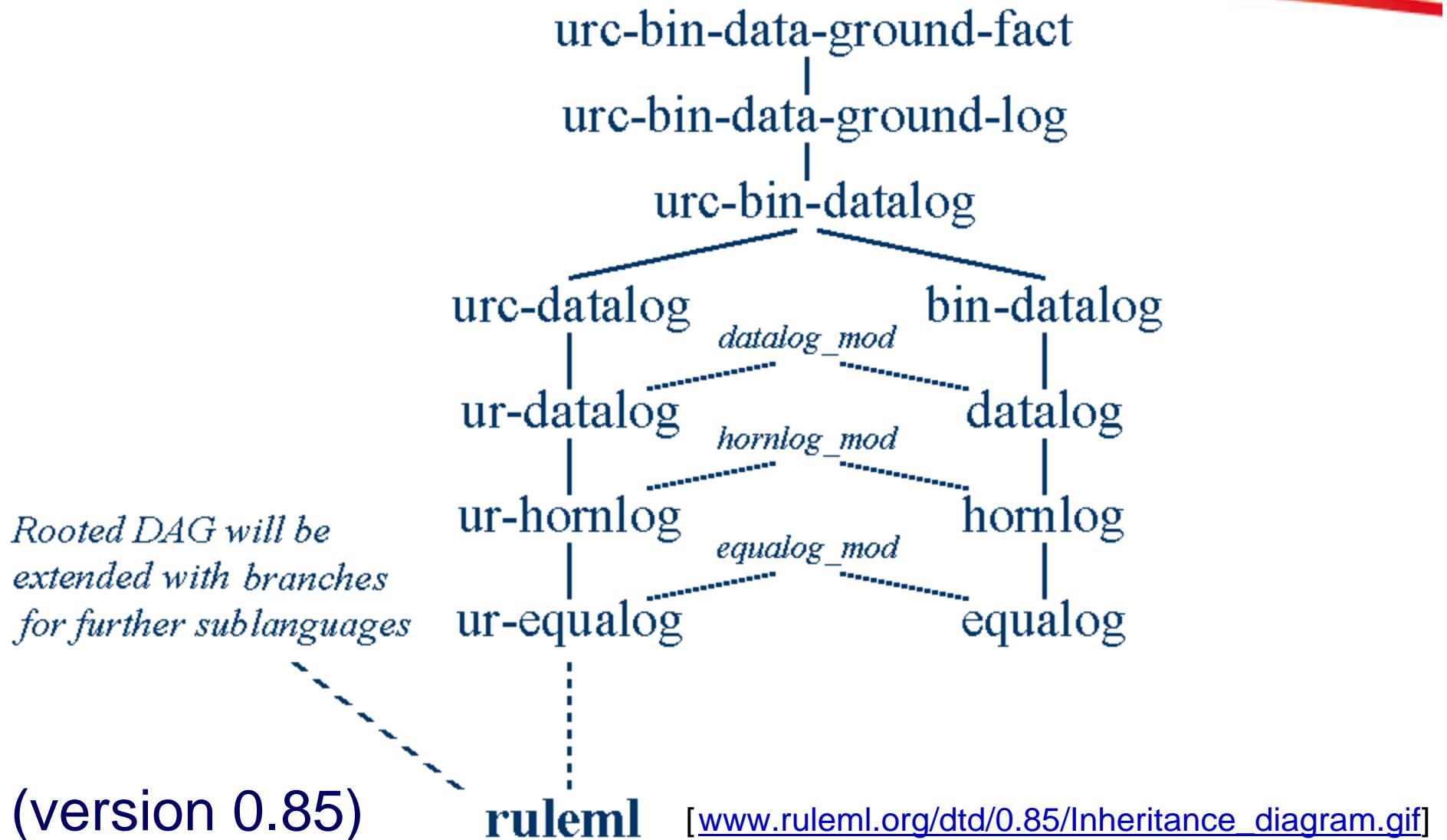
- however, v. 0.8 inheritance less than optimal
  - counter-intuitive
  - “inconsistent”
  - inefficient

# DTDs - Inheritance Diagram (v. 0.8)



## DTDs - Remodularization (v. 0.85)

- single root with two distinct branches (simplicity)
  - far more intuitive
  - simplified tree
  - inverted
- inheritance in one direction only (consistency)
  - obvious super/subclass relationships
  - each node inherits from node directly above it
- non-redundant (efficiency)
  - use of mods for changes affecting multiple DTDs



## DTDs - Inheritance with Entities

- DTDs have limited support for modularity
- can still be accomplished with macro-like entities:  
(note similarity to predefined ones in HTML)

e.g. `<!ENTITY copy "Copyright 2003. All rights reserved.">`  
`<!-- using &copy; in document will print text -->`

- usable only within DTD: parameter entities

e.g. `<!ENTITY % author "John Doe">`

- useful as a roundabout way to “inherit” the contents of another file

e.g. `<!ENTITY % datalog_include SYSTEM "datalog.dtd">`  
`%datalog_include;`

## DTDs - Overriding

- inclusion of other documents isn't enough
  - what about overriding?
    - version 0.8 used INCLUDE/IGNORE
      - e.g. to change metarole `_r` from

```
<!ELEMENT _r (ind)> (in urcbindataground fact)
```

to

```
<!ELEMENT _r (ind | var)> (in urcbindatalog),
```

declaration of `_r` would be IGNOREd in datalog, then declared separately in hornlog
  - version 0.85 uses content model-based approach

## DTDs - Content Models

- create a parameter entity for each element's content model

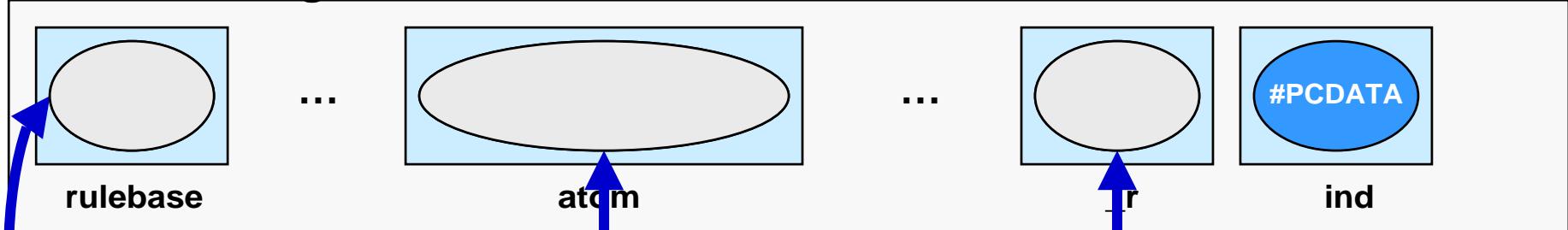
e.g. `<!ENTITY % ind.content "(#PCDATA)">  
<!ELEMENT ind %ind.content;*>`

- subclasses overwrite param. entity with new content model
  - elements/attributes can't overwrite one another (only entities can)
  - analogous to re-assigning global variables

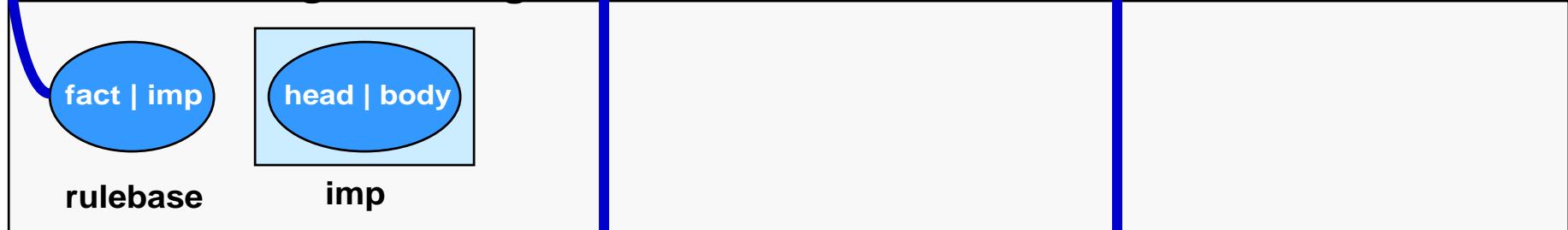
e.g. `<!-- in urcbindatagroundfact.dtd -->  
<!ENTITY % _r.content "(ind)">  
<!ELEMENT _r %_r.content;*>`

`<!-- in urcbindatalog.dtd -->  
<!ENTITY % _r.content "(ind | var)">`

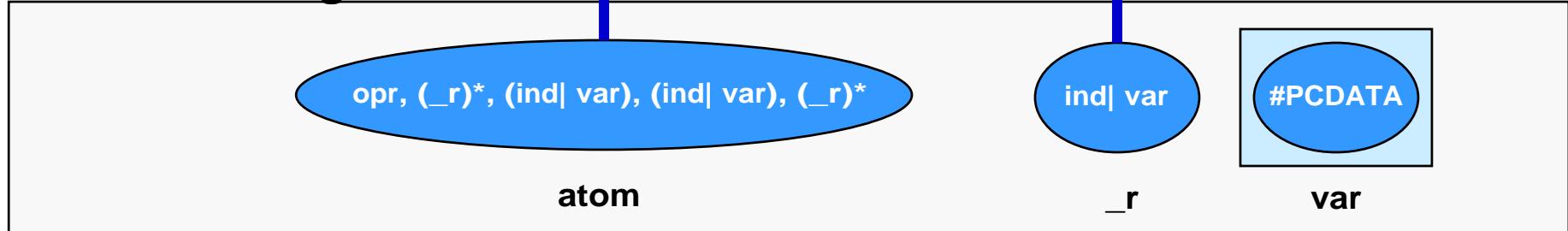
## urc-bin-data-ground-fact



## urc-bin-data-ground-log



## urc-bin-datalog



## DTDs - Demo

DTD directory listing [[www.ruleml.org/dtd/0.85/](http://www.ruleml.org/dtd/0.85/)]

DTD example directory [[www.ruleml.org/exa/0.85/](http://www.ruleml.org/exa/0.85/)]

Online validator [[www.stg.brown.edu/service/xmlvalid/](http://www.stg.brown.edu/service/xmlvalid/)]

# XML Schema Definition (XSD)

- DTDs are limited
  - not XML syntax
  - no constraints on character data
  - “brute force” inheritance
- XML Schema is better ...
  - XML syntax
  - datatypes
  - namespaces
- ... but not perfect
  - modularity mechanisms are vague
  - very complex and verbose

## XSD - Content Models

- content model-based approach also works with XSD
  - instead of parameter entities, use groups  
e.g. `<!ENTITY % _r.content "&(ind)">`  
`<!ELEMENT _r %_r.content;>`

becomes

```
<xs:attributeGroup name="_r.attlist"/>
<xs:group name="_r.content">
    <xs:sequence>
        <xs:element ref="ind"/>
    </xs:sequence>
</xs:group>
<xs:complexType name="_r.type" mixed="true">
    <xs:group ref="_r.content"/>
    <xs:attributeGroup ref="_r.attlist"/>
</xs:complexType>
<xs:element name="_r" type="_r.type"/>
```

## XSD - Inheritance

- no need for workarounds in XSD
  - <redefine> makes changes and includes everything else  
e.g. `<!ENTITY % _r.content "(ind | var)">`  
`<!ENTITY % include SYSTEM "urcbindatagroundlog.dtd">`  
`%include;`

becomes

```
<xs:redefine schemaLocation="urcbindatagroundlog.xsd">
  <xs:group name="_r.content">
    <xs:choice>
      <xs:group ref="_r.content"/>
      <xs:element ref="var"/>
    </xs:choice>
  </xs:group>
</xs:redefine>
```

## XSD - Demo

XSD directory listing [[www.ruleml.org/xsd/0.85/](http://www.ruleml.org/xsd/0.85/)]

XSD example directory [[www.ruleml.org/exa/0.85/](http://www.ruleml.org/exa/0.85/)]

Online validator [[www.w3.org/2001/03/webdata/xsv](http://www.w3.org/2001/03/webdata/xsv)]

## Steering Committee

- presented to RuleML Steering Committee during teleconference
  - Wednesday, November 5th, 2003 2:00pm AST
- Committee members:
  - Harold Boley (CA)
  - Mike Dean (USA)
  - Andreas Eberhart (DE)
  - Benjamin Grosof (USA)
  - Duncan Johnston-Watt (UK)
  - Steve Ross-Talbot (UK)
  - Bruce Spencer (CA)
  - Said Tabet (USA)
  - Gerd Wagner (NL)
- work was approved

## Future Work

- existing issues
    - negation
      - classical/strong
      - as failure
    - and/or nesting
    - transformation rules, reaction rules
    - guarded Horn Logic (suggested by Wolfgang Nejdl, U Hannover)
    - abstract syntax
  - further suggestions from Benjamin Grosof
    - SCLP (Situated Corteous Logic Programs)
- These have since been implemented:  
[[www.ruleml.org/dtd/0.85/](http://www.ruleml.org/dtd/0.85/)]  
[[www.ruleml.org/xsd/0.85/](http://www.ruleml.org/xsd/0.85/)]

## Questions/ Comments?

- References
  - Modularization of XHTML (with DTDs) (W3C Rec.)  
[[www.w3.org/TR/xhtml-modularization](http://www.w3.org/TR/xhtml-modularization)]
  - Modularization of XHTML (with XSD) (W3C WD)  
[[www.w3.org/TR/xhtml-m12n-schema](http://www.w3.org/TR/xhtml-m12n-schema)]
  - Rule Markup Initiative [[www.ruleml.org](http://www.ruleml.org)]