Highly-Modular Relax NG Schemas for Customized Access to RuleML Knowledge Bases

Tara Athan[1], Harold Boley[2]

[1]Athan Services, W. Lafayette, Indiana

[2]Institute for Information Technology, National Research Council; Faculty of Computer Science, University of New Brunswick, Canada

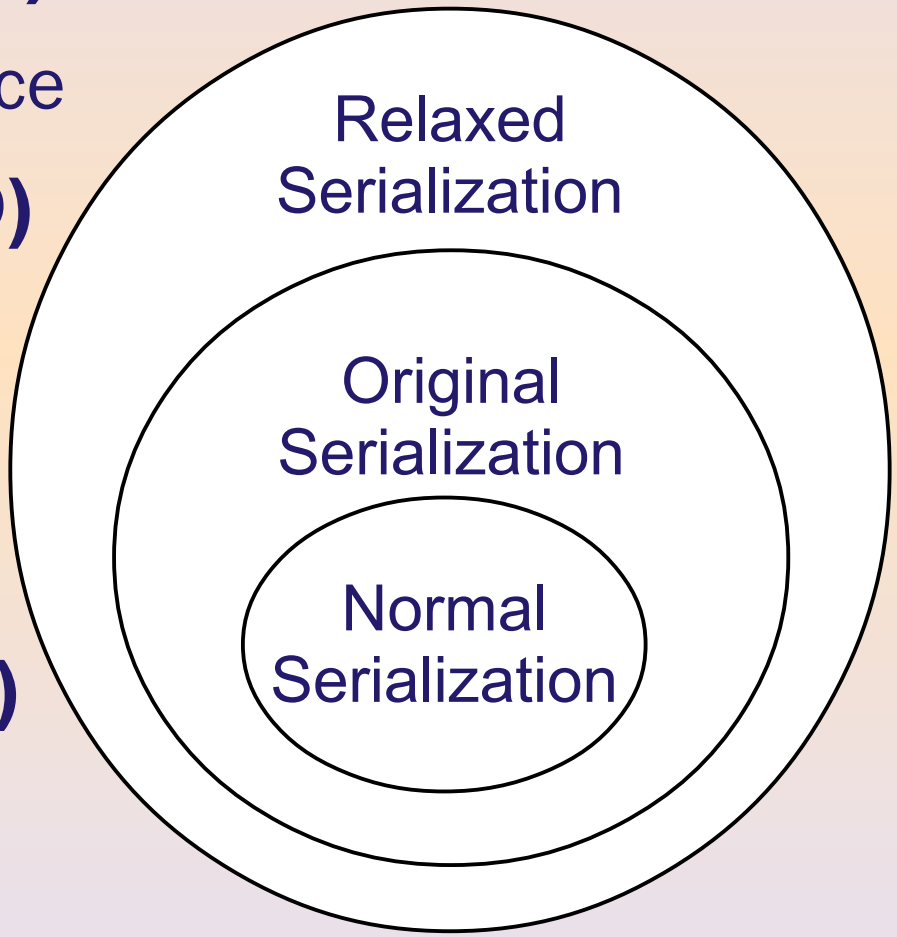
OMG Technical Meeting, Ontology PSIG
Cambridge, MA, 21 June 2012

Re-conceptualization and Re-engineering: Goals

- **Language Extensions**
  - Decreased positional sensitivity
  - More flexibility in defining sublanguages
- **Greater Reliability**
- **Greater Automation**
  - Testing, documentation, conversion

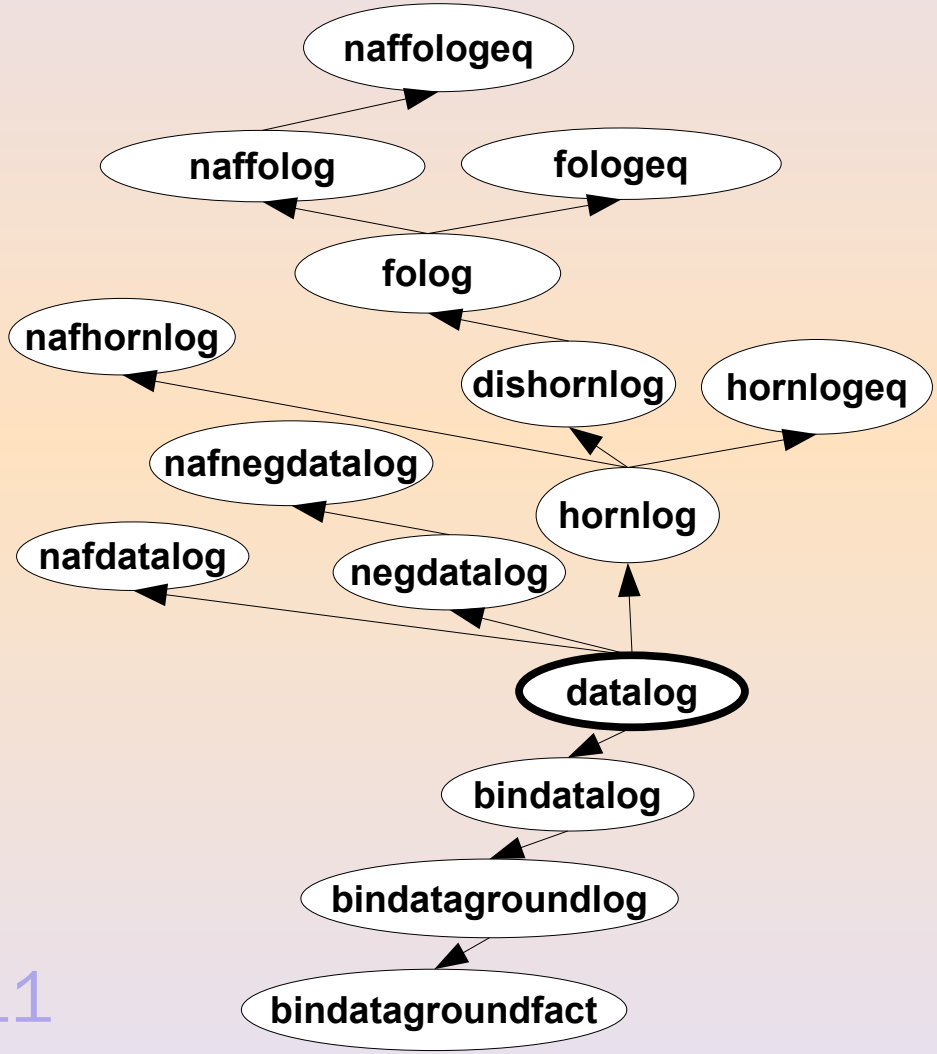
Relationship of RNC and XSD: Syntactic Inclusion

- **Relaxed Serialization (RNC)**
  - More positional independence
- **Original Serialization (XSD)**
  - Optional Stripes
  - Some positional
  - independence
- **Normal Serialization (RNC)**
  - Fully-striped
  - Canonical Position



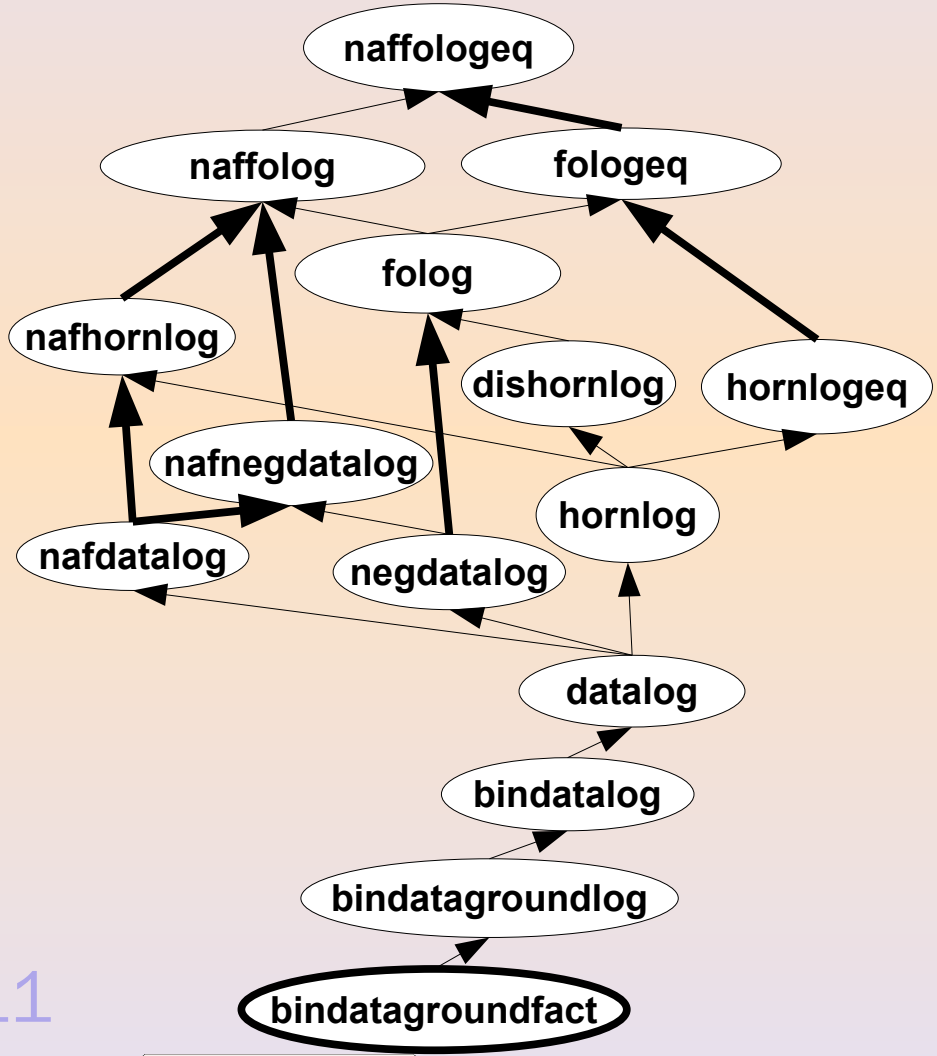
Modularization: "Original Fifteen" (non-SWSL)

- **RuleML XSDs use directed tree-based modularization**
- **RuleML Relax NG uses lattices**
- **Lattice vertices can be assigned codes**
  - Bitwise-dominance indicates containment
    1111 = 001111 < 101111



Modularization:
Original Fifteen FOL

- RuleML XSDs use directed tree-based modularization
- RuleML Relax NG uses lattices
- Lattice vertices can be assigned codes
  - Bitwise-dominance indicates containment
    1111 = 001111 < 101111



Modular sYNtax confiGurator (MYNG) http://ruleml.org/1.0/myng/

Selection Form

Instructions

Make a selection from the form below, then click "Refresh Schema" to update the Schema URL. The main module is also displayed below the form. To reset the form to the default (supremum) values, click "Reset Form".

Reset Form | Refresh Schema

Schema URL = http://ruleml.org/1.0/relaxng/schema_rnc.php?backbone=x3f&default=x7&termseq=x7&lng=x1&propo=x3ff&implies=x7&terms=xf3f&quant=x7&expr=xf&serial=xf

MYNG
Customization Form Part 1

Expressivity "Backbone" (Check One)
- ○ Atomic Formulas
- ○ Ground Fact Logic
- ○ Datalog
- ○ Horn Logic
- ○ Disjunctive

Treatment of Attributes With Default Values (Check One)
- ○ Required to be Absent
- ○ Required to be Present
- ◉ Optional

Term Sequences: Number of Terms (Check One)
- ○ None
- ○ Binary (Zero or Two)
- ◉ Polyadic (Zero or More)

Language (Check One)
- ◉ English Abbreviated Names
- ○ English Long Names
- ○ French Long Names

Serialization Options (Check Zero or More)
- ☑ Unordered Groups
- ☑ Stripe-Skipping
- ☑ Explicit Datatyping
- ☑ Schema Location Attribute

MYNG
Customization Form Part 2

Propositional Options (Check Zero or More)
- ☑ IRIs
- ☑ Rulebases
- ☑ Entailments
- ☑ Degree of Uncertainty
- ☑ Strong Negation
- ☑ Weak Negation
- (Negation

Implication Options (Check Zero or More)
- ☑ Equivalences
- ☑ Inference Direction
- ☑ Non-Material

Term Options (Check Zero or More)
- ☑ Object Identifiers
- ☑ Slots
- ☑ Slot Cardinality
- ☑ Slot Weight
- ☑ Equations
- ☑ Oriented Equations

Quantification Options (Check Zero or More)
- ☑ Implicit Closure
- ☑ Slotted Rest Variables
- ☐ Positional Rest Variables

Expression Options (Check Zero or More)
- ☐ Generalized Lists
- ☐ Set-valued Expressions
- ☐ Interpreted Expressions

MYNG
Usage: Online or Download

Usage

The Schema URL may be used directly for online validation - copy and paste as required by the validator. For a demonstration of validation using the online service Validator.nu, see How to Validate with the RuleML Parameterized Relax NG Schema. Some scripts and processing instructions may require that the character "&" be replaced by "&". Clicking on the Schema URL downloads a copy of the schema driver into a file named "custom_driver.rnc". To use the schema driver locally (offline), a local copy of the modules directory is also necessary - this may be downloaded as a zip archive from the RuleML 1.0 Relax NG Directory.

MYNG
Customized Driver Display

```
 start = Node.choice | edge.choice
#
# ROOT NODE AND PERFORMATIVES INCLUDED
#
include "modules/performative_expansion_module.rnc" inherit = ruleml {start |= notAllowed}
#
# ATOMIC FORMULAS INCLUDED
#
include "modules/atom_expansion_module.rnc" inherit = ruleml {start |= notAllowed}
#
# CONJUNCTIONS AND DISJUNCTIONS INCLUDED
#
include "modules/andor_expansion_module.rnc" inherit = ruleml {start |= notAllowed}
#
# IMPLICATIONS INCLUDED
#
include "modules/implication_expansion_module.rnc" inherit = ruleml {start |= notAllowed}
#
# QUANTIFICATION OVER VARIABLES INCLUDED
#
include "modules/quantification_expansion_module.rnc" inherit = ruleml {start |= notAllowed}
#
# ATTRIBUTES WITH DEFAULT VALUES ARE INITIALIZED
#
include "modules/default_inf_expansion_module.rnc" inherit = ruleml {start |= notAllowed}
#
# ATTRIBUTES WITH DEFAULT VALUES ARE ABSENT OR OPTIONAL
```

RNC as Content Model

- **XSD**

```
<xs:element name="RuleML">
  <xs:complexType>
    <xs:sequence>
      <xs:element minOccurs="0"
         ref="ruleml:oid"/>
      <xs:choice minOccurs="0"
         maxOccurs="unbounded">
        <xs:element
         ref="ruleml:act"/>
        <xs:element
         ref="ruleml:Assert"/>
        <xs:element
         ref="ruleml:Retract"/>
        <xs:element
         ref="ruleml:Query"/>
      </xs:choice> ...
```

- **RNC**

```
RuleML = element
  RuleML {
    oid?,
    ( act
      | Assert
      | Retract
      | Query)* }
```

Serializations Compared

- **RNC normal**

```
Atom = element Atom {
  attribute closure {
    "universal"
    | "existential" }?,
  oid?, degree?,
  op,
  arg*,
  repo?,
  slot*,
  resl?
}
```

- **RNC relaxed**

```
Atom = element Atom {
  attribute closure {
    "universal"
    | "existential" }?,
  (oid? & degree?),
  ((op|Rel) &
  (arg|arg.content)* &
  repo? &
  slot* &
  resl?)
}
```

Syntactic Monotonicity

- **Definition:**
  - Grammar containment implies syntactic containment
- **Relax NG (like XSD) is not monotonic**
  - redefinition
  - interleave combine "&="

- xy.rnc

```
start = x
x = element x{ x.main }
x.main = y?
y = element y{ text }
```

- xy_redefine.rnc

```
include xy.rnc {
  x.main = y+ }
```

- xy_interleave.rnc

```
include xy.rnc
x.main &= y
```

Schema Design Pattern: Sufficient to Achieve Monotonicity

- **Segregated Names**
  - Choice combine
  - No combine
  - Interleave combine
    - &= empty
    - &= ...?
    - &= ...*
- **Joins by union, not redefinition**

```
Equal-node.choice |=
  Equal.Node.def

Equal.Node.def =
  element Equal {
    (Equal-datt.choice &
     reEqual.attlist),
    Equal.header, Equal.main}

Equal.header &=
  SimpleFormula.header?

Equal.main |=
  leftSide-edge.choice,
  rightSide-edge.choice
```

Expressivity of Schema Design Pattern

- **Any valid RNC schema can be expressed using the schema design pattern**
- **Any language lattice where each language has a valid RNC schema can be modularized using the schema design pattern**

```
RuleML =
   element RuleML
{...}
act =
   element act {...}
...
```

Measurable Outcomes: Increased customizability

- **Over fifty freely combinable modules**
  - Decoupling elements such as <Atom>
- **More than $2^{50} > 10^{15}$ grammars**
- **generating an estimated 300,000 different (and meaningful) languages.**