Design and Implementation of Highly Modular Schemas for XML: Customization of RuleML in Relax NG

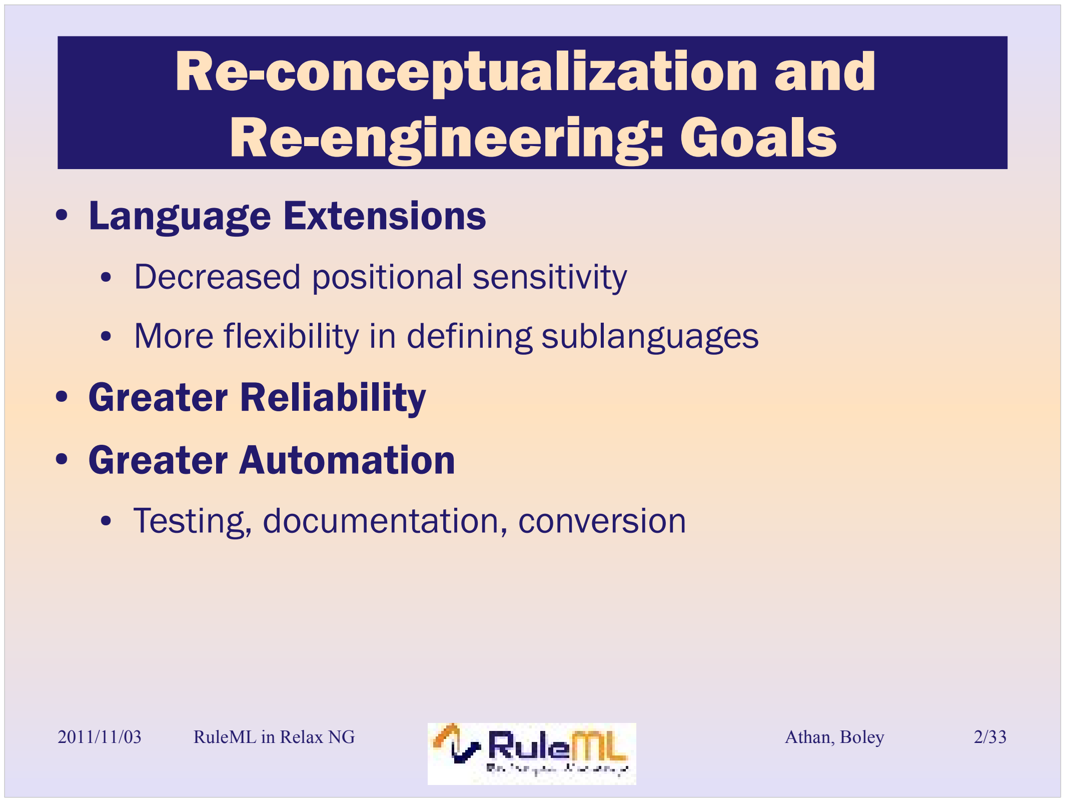
RuleML2011@BRF Fort Lauderdale, Florida

Tara Athan[1], Harold Boley[2]

[1]Athan Services, Ukiah, California

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

3 November 2011



Re-conceptualization and Re-engineering: Goals

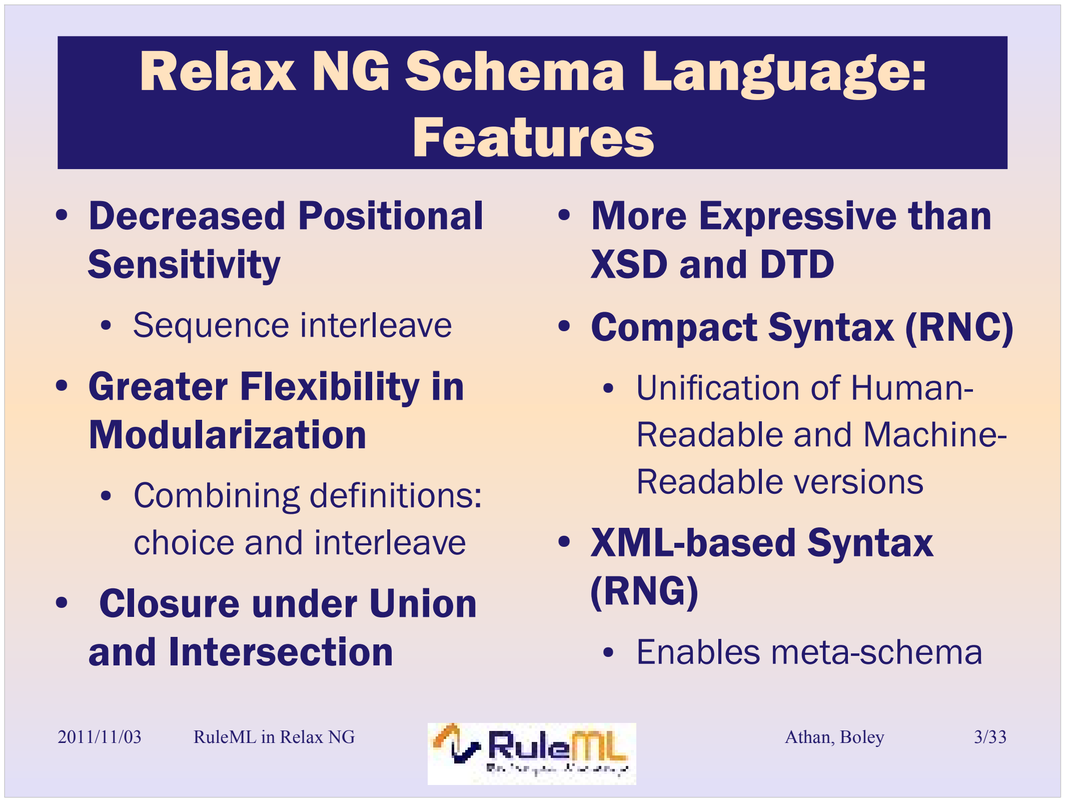
- **Language Extensions**
  - Decreased positional sensitivity
  - More flexibility in defining sublanguages
- **Greater Reliability**
- **Greater Automation**
  - Testing, documentation, conversion

RuleML is a family of languages for Web rule interchange that was originally specified in Document Type Definitions (DTDs), then switched to XML Schema Definition Language (XSD) schemas.

Here we present a re-engineering and re-conceptualization of the non-SWSL portion of the Derivation Rules subfamily of RuleML in the Relax NG Compact (RNC) schema syntax.

The goals of this effort are

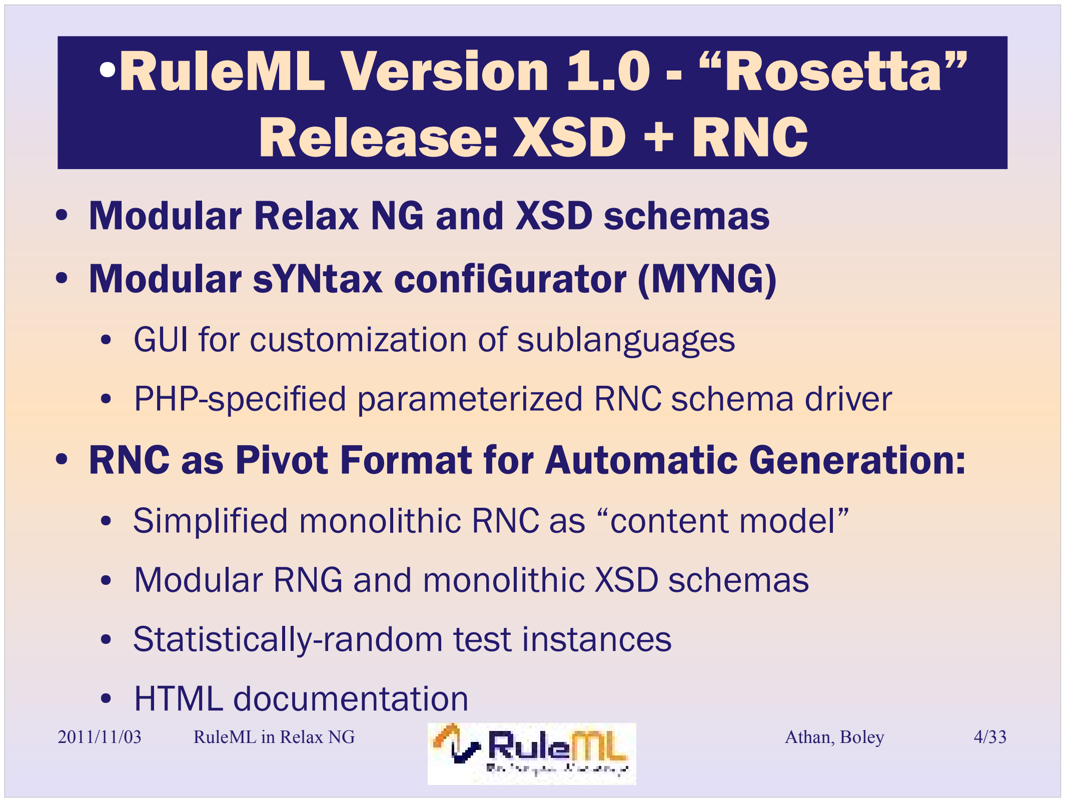
* RuleML language extensions, including decreased positional sensitivity and greater flexibility in defining sublanguages
* greater reliability.
* Greater automation via the use of web applications (preferred) or desktop software.



Relax NG Schema Language: Features

- **Decreased Positional Sensitivity**
  - Sequence interleave
- **Greater Flexibility in Modularization**
  - Combining definitions: choice and interleave
- **Closure under Union and Intersection**
- **More Expressive than XSD and DTD**
- **Compact Syntax (RNC)**
  - Unification of Human-Readable and Machine-Readable versions
- **XML-based Syntax (RNG)**
  - Enables meta-schema

There are several features of Relax NG that led us to select it over XSD as the pivot format for our modular schemas.
* Decreased positional sensitivity is introduced through the sequence interleave operator
* Definitions may be combined as either a choice or an interleave, allowing greater flexibility in modularization
* Due to the basis of Relax NG in hedge automaton theory, it has closure under Union and Intersection.
* Relax NG is more expressive than XSD
* The compact syntax is both human and machine-readable
* The XML-based syntax may be defined as a meta-schema in RNC



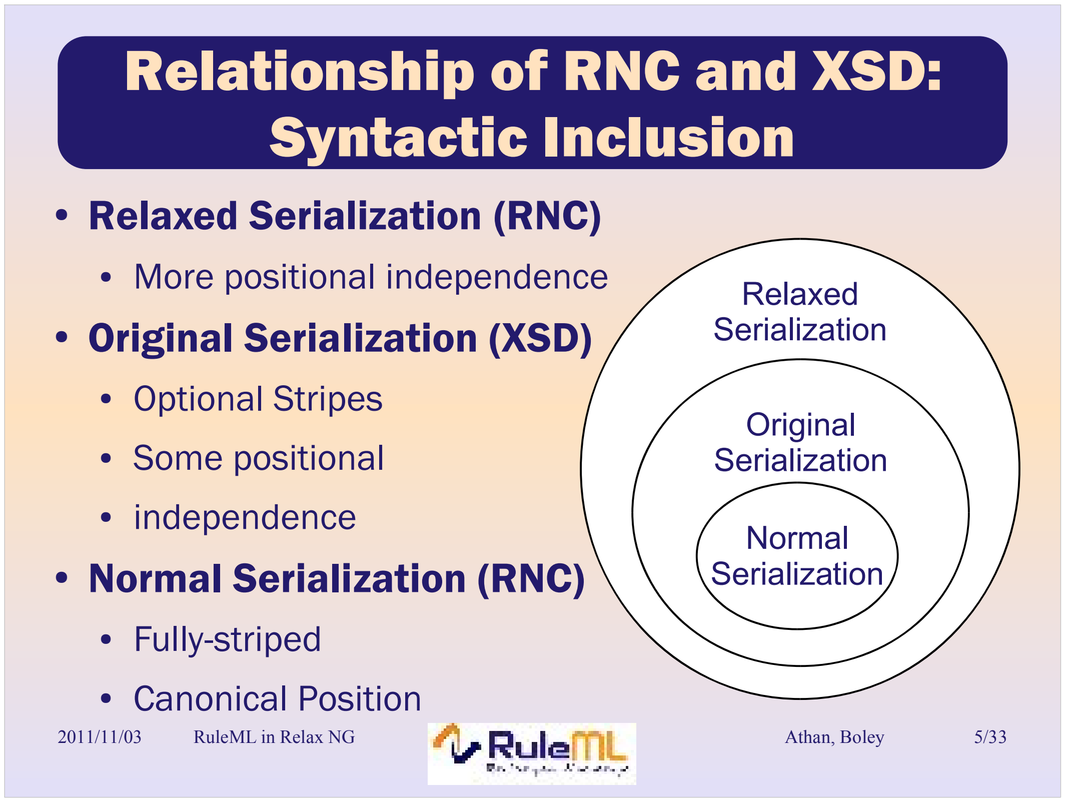
•RuleML Version 1.0 - “Rosetta” Release: XSD + RNC

- **Modular Relax NG and XSD schemas**
- **Modular sYNtax confiGurator (MYNG)**
  - GUI for customization of sublanguages
  - PHP-specified parameterized RNC schema driver
- **RNC as Pivot Format for Automatic Generation:**
  - Simplified monolithic RNC as “content model”
  - Modular RNG and monolithic XSD schemas
  - Statistically-random test instances
  - HTML documentation

The Version 1.0 release is a "Rosetta" release, allowing comparison of the XSD schemas as originally developed, and the syntax they define, with the Relax NG schemas and its corresponding syntax.

Because of the large number of sublanguages available in the re-engineered approach, we developed MYNG, consisting of a GUI front end to a PHP script, for custom configuration of RuleML sublanguages through on-the-fly building of a schema driver file that includes a selection of modules.

The RNC schemas act as a pivot format in that it is possible to automatically generate from them schemas in other formats (modular RNC, monolithic RNC and XSD) as well as statistically random instances and HTML documentation.



Relationship of RNC and XSD: Syntactic Inclusion

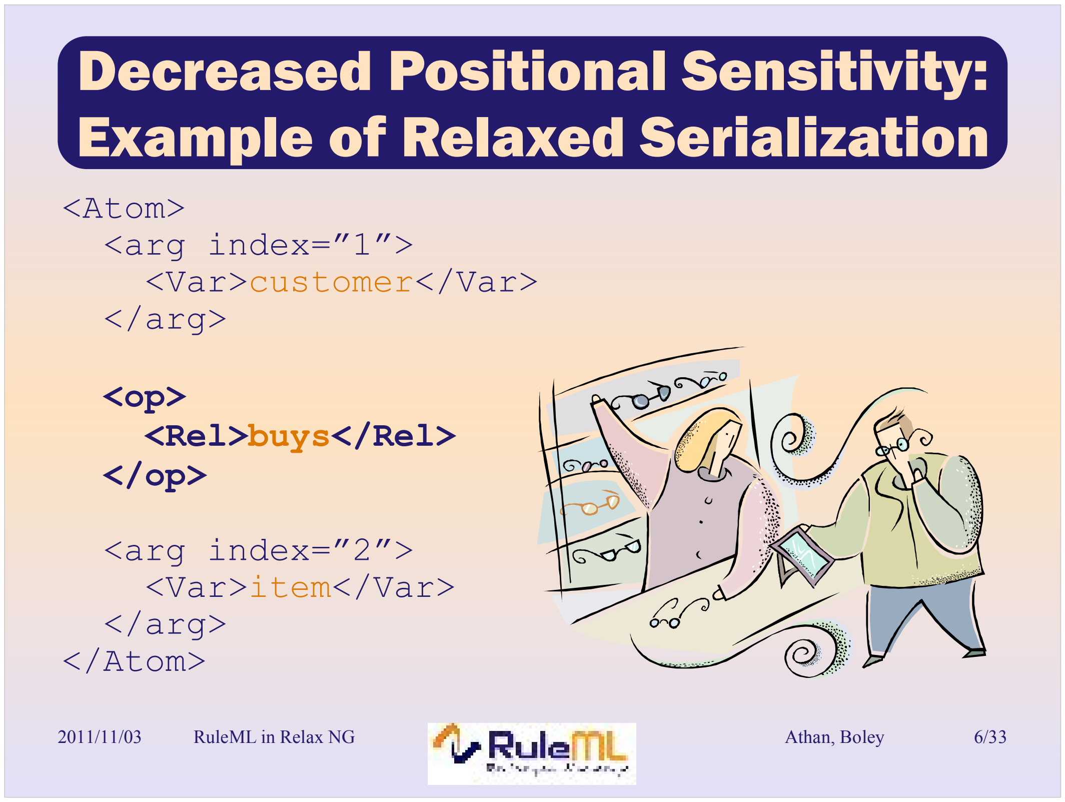
- **Relaxed Serialization (RNC)**
  - More positional independence
- **Original Serialization (XSD)**
  - Optional Stripes
  - Some positional
  - independence
- **Normal Serialization (RNC)**
  - Fully-striped
  - Canonical Position

Relaxed Serialization
Original Serialization
Normal Serialization

In the RuleML Version 1.0 Rosetta Release, each of the "Original 15" sublanguages, as defined in the corresponding XSD schema, is bracketed by a pair of sublanguages defined in Relax NG.

The most tolerant sublanguage is called the "relaxed serialization", and has more positional independence than the original sublanguage.

The most restrictive sublanguage is called the "normal serialzation". It is a fully-striped syntax, and requires elements to occur in a canonical ordering.



Decreased Positional Sensitivity: Example of Relaxed Serialization

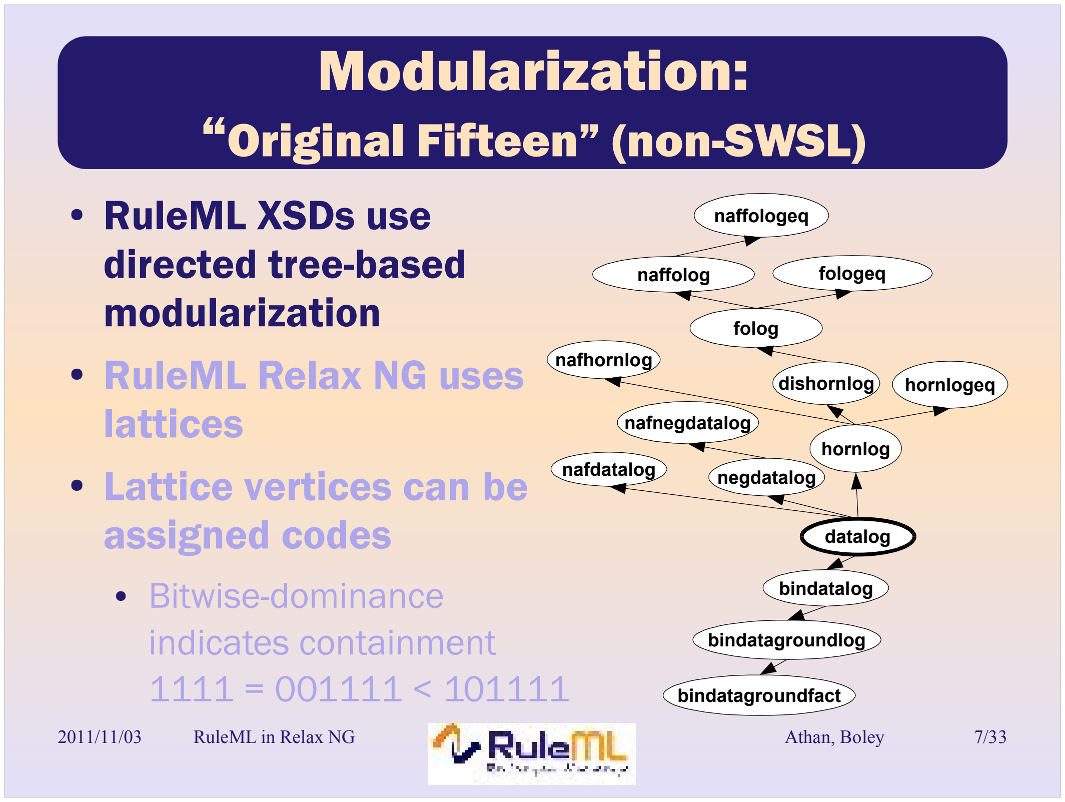
```
<Atom>
  <arg index="1">
    <Var>customer</Var>
  </arg>

  <op>
    <Rel>buys</Rel>
  </op>

  <arg index="2">
    <Var>item</Var>
  </arg>
</Atom>
```

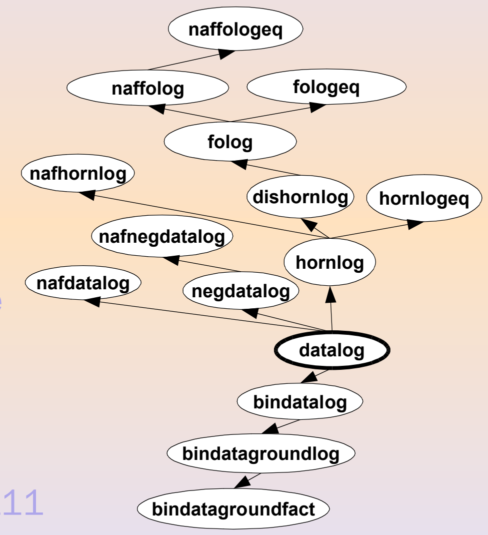
The Relax NG schemas introduce only a few new features to the language itself. A fragment of RuleML with infix operator notation is shown. This formula is allowed by the relaxed serialization, but was not a valid formula according to the XSD schemas.

Infix and postfix operator notation allows relations to be used in a natural language ordering, as in "A customer buys an item."



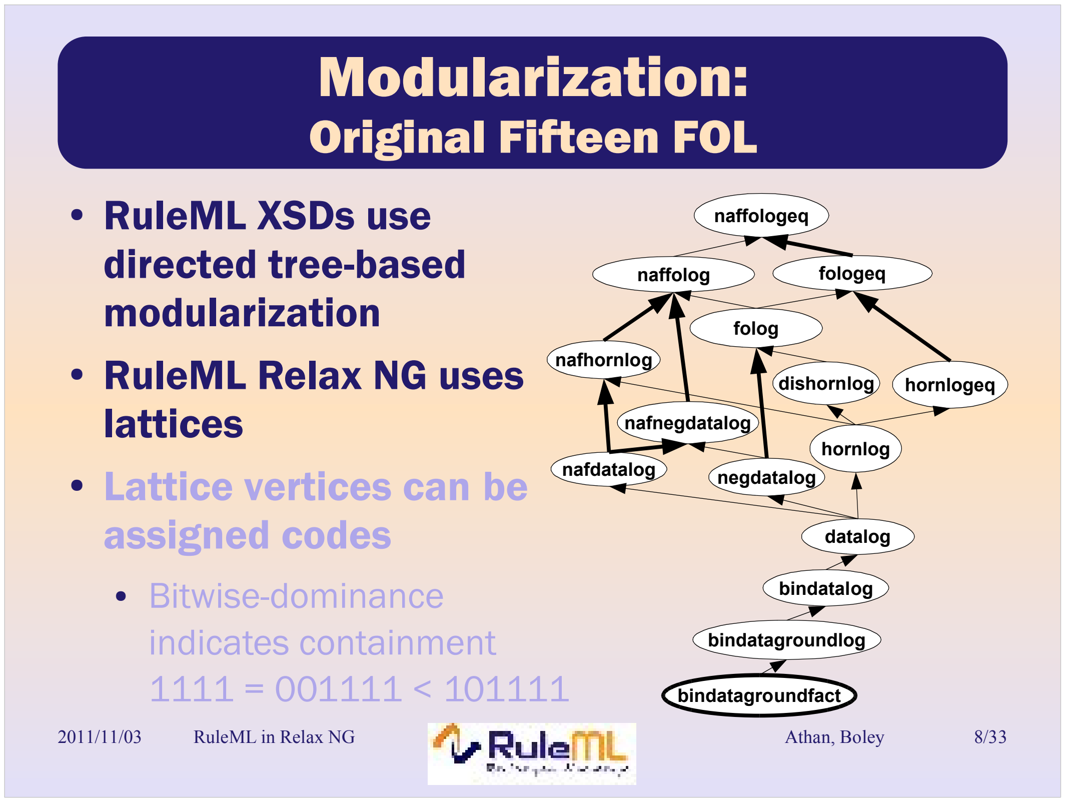
Modularization: "Original Fifteen" (non-SWSL)

- **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



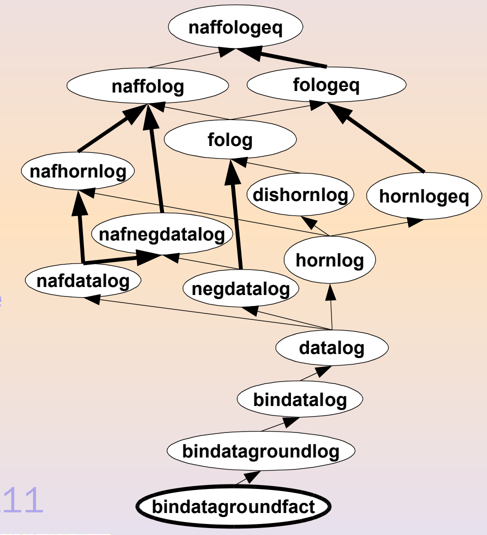
The RuleML XSD modularization approach has been based on a tree.

The datalog schema is the root of the tree. An arrow indicates a schema obtained by merging, possibly with redefinition, the schema at the tail of the arrow with one or more modules (that are not shown).



Modularization: Original Fifteen FOL

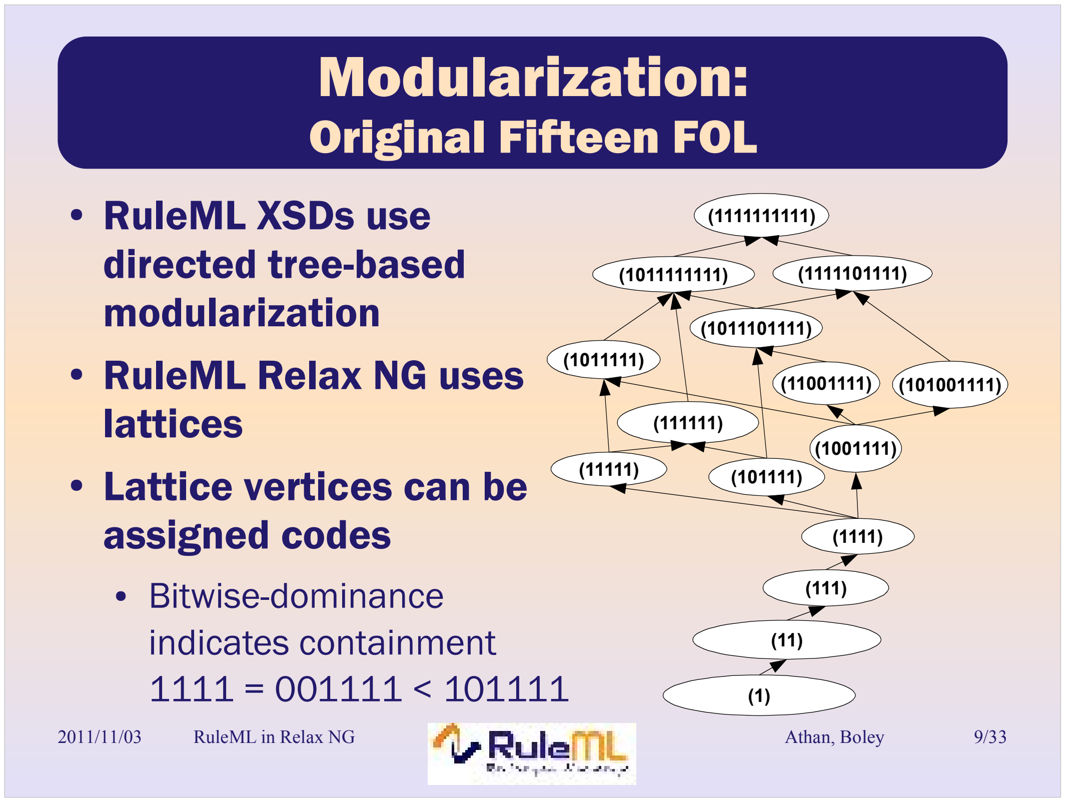
- **RuleML XSDs use directed tree-based modularization**
- **RuleML Relax NG uses lattices**
- **Lattice vertices can be assigned codes**
  - Bitwise-dominance indicates containment 1111 = 001111 < 101111



The RuleML Relax NG modularization approach is based on a lattice. The original fifteen languages can be represented graphically as a lattice by adding a few arrows.

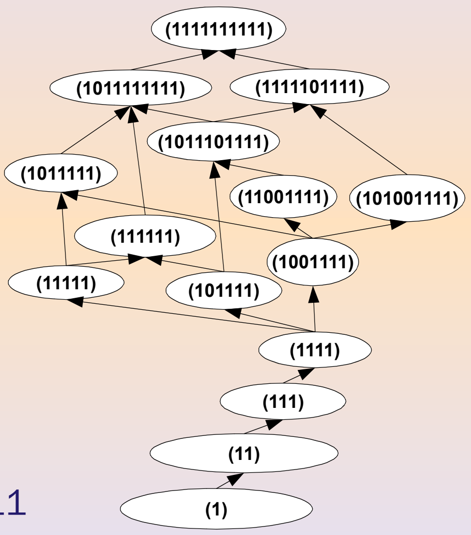
The bottom (infimum) of the lattice is the bindatagroundfact sublanguage, so the direction of the arrows in the lowest branch is reversed from the previous slide. Now the arrows are all consistent with the partial order of containment, pointing towards the containing sublanguage.

The partial order represented here is based on syntactic containment.
Every valid instance in the contained sublanguage is also valid in the containing sublanguage. Semantics is defined in the top language, so that the meaning of a statement is never dependent on what sublanguage it belongs to.



Modularization:
Original Fifteen FOL

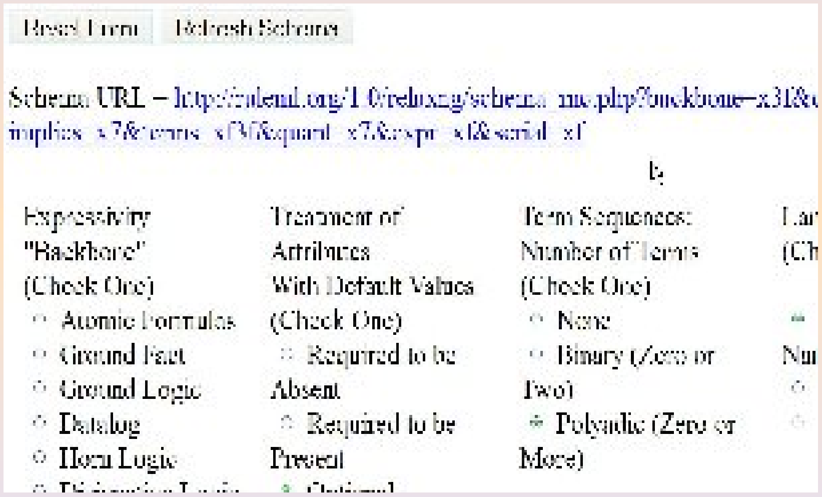
- **RuleML XSDs use directed tree-based modularization**
- **RuleML Relax NG uses lattices**
- **Lattice vertices can be assigned codes**
  - Bitwise-dominance indicates containment 1111 = 001111 < 101111



The lattice structure, arising from the partial order containment (syntactic and semantic) can be encoded as a binary number, where order is based on bit-wise dominance.

For example, to compare the codes 1111 (datalog) and 101111 (negdatalog), we first pad with zeros on the left
1111 = 001111
and then compare bit-wise
001111 < 101111

Modular sYNtax confiGurator http://ruleml.org/1.0/myng/


Reset Form | Refresh Schema

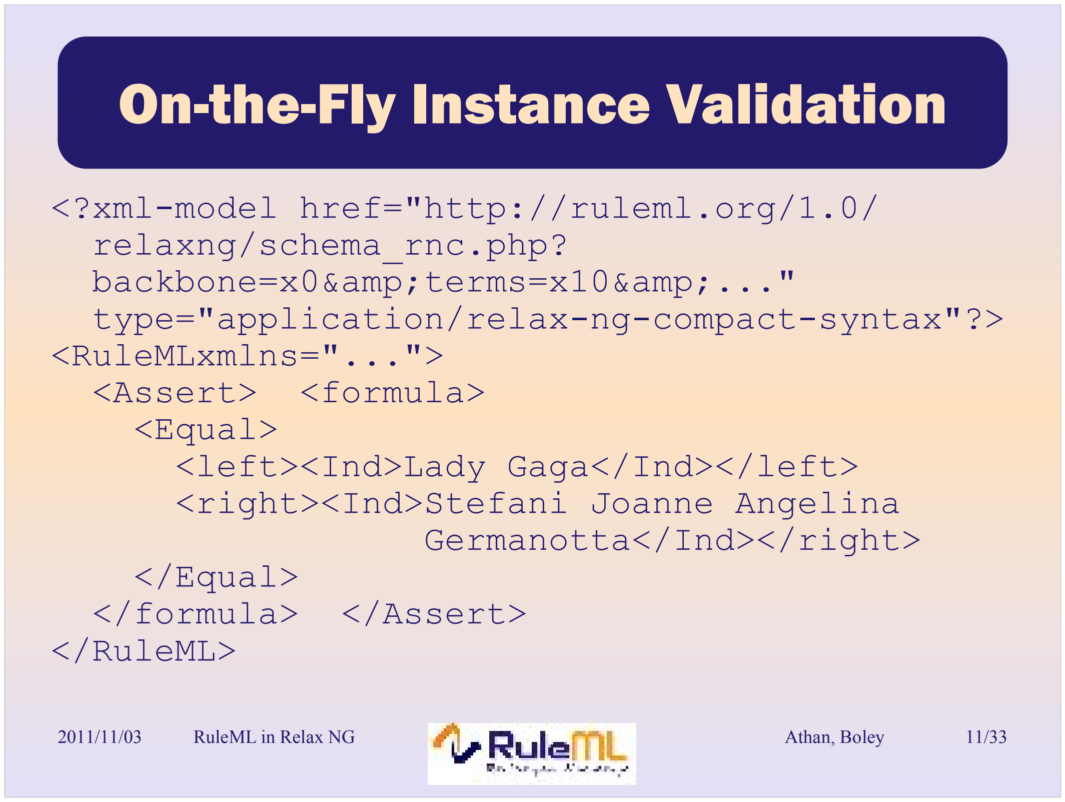
Schema URL – http://ruleml.org/1.0/relaxng/schema_rnc.php?backbone=x3f&
implies_x7&terms_xf3f&quant_x7&expr_x1&serial_xf

Expressivity "Backbone" (Check One)	Treatment of Attributes With Default Values (Check One)	Term Sequences: Number of Terms (Check One)	Lan (Ch
○ Atomic Formulas	○ Required to be Absent	○ None	●
○ Ground Fact	○ Required to be Present	○ Binary (Zero or Two)	Nu
○ Ground Logic	[✓ ● Optional]	[✓ ● Polyadic (Zero or More)]	○
○ Datalog			○
○ Horn Logic			

A java script-driven web application (http://ruleml.org/1.0/myng), the Modular sYNtax confiGurator (MYNG), pronounce either "ming" or "my NG", has been created to assist with managing the large number of new sublanguages available through the lattice-based modularization approach.

The options are organized into facets of semantically related features. For each facet, the user's selection is converted to the appropriate hexadecimal code for the query string.

The Refresh Schema button updates the URL according to the user's selections, and also displays the text of the driver schema (not shown).
The URL can be used to download the driver schema, or for online validation, as will be shown during the demonstration session.

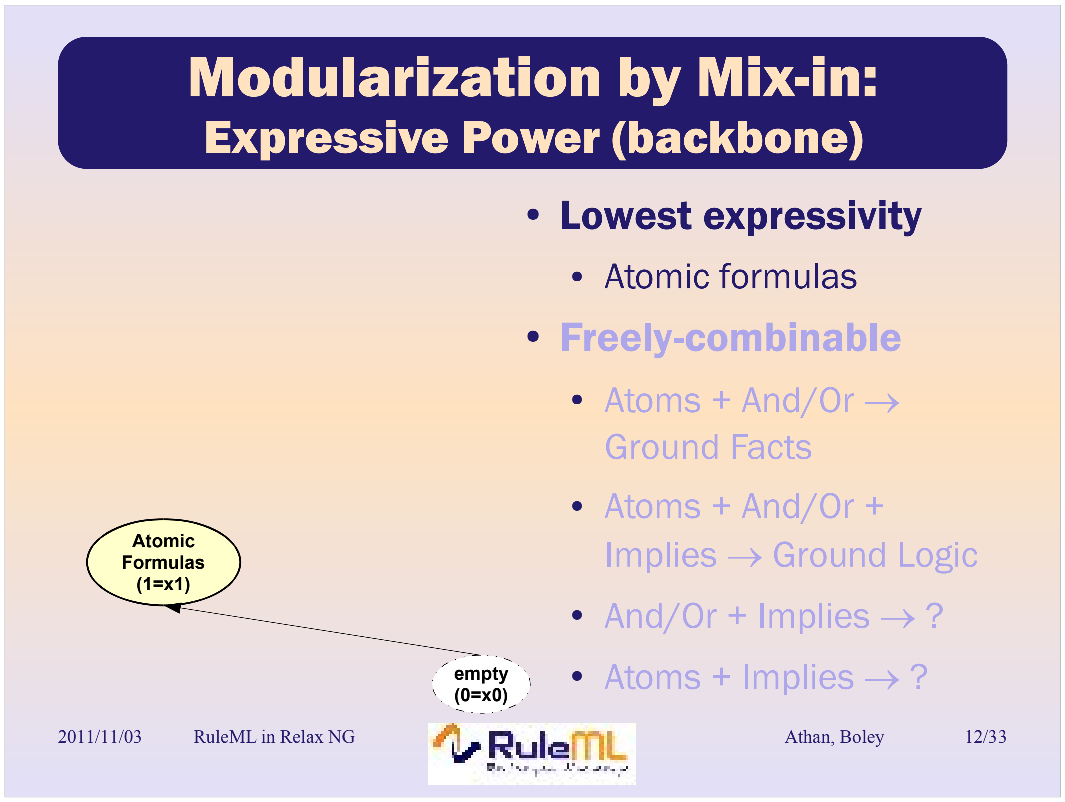


On-the-Fly Instance Validation

```
<?xml-model href="http://ruleml.org/1.0/
  relaxng/schema_rnc.php?
  backbone=x0&amp;terms=x10&amp;..."
  type="application/relax-ng-compact-syntax"?>
<RuleMLxmlns="...">
  <Assert>  <formula>
    <Equal>
      <left><Ind>Lady Gaga</Ind></left>
      <right><Ind>Stefani Joanne Angelina
                  Germanotta</Ind></right>
    </Equal>
  </formula>  </Assert>
</RuleML>
```

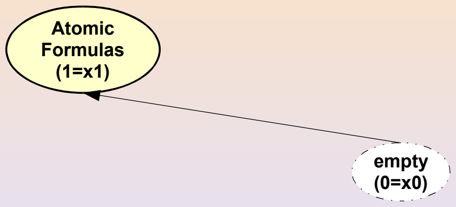
The parametrized schema URL may appear as an attribute of an xml-model processing instruction, for use in validation by xml-model processing software, such as oXygen (http://oxygenxml.com)

An example is shown of a minimal language (backbone=x0) with only atomic, nullary facts that includes equations (terms=x10). This is a sublanguage that is not represented among the "Original Fifteen".

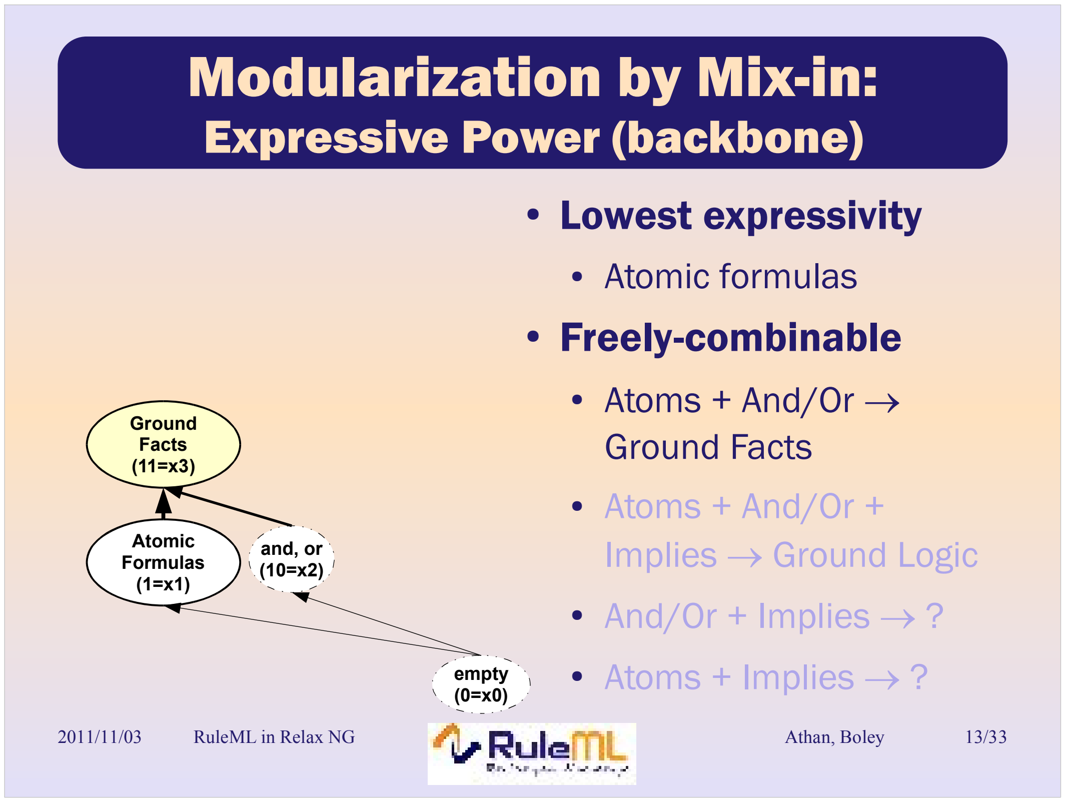


Modularization by Mix-in: Expressive Power (backbone)

- **Lowest expressivity**
  - Atomic formulas
- **Freely-combinable**
  - Atoms + And/Or → Ground Facts
  - Atoms + And/Or + Implies → Ground Logic
  - And/Or + Implies → ?
  - Atoms + Implies → ?

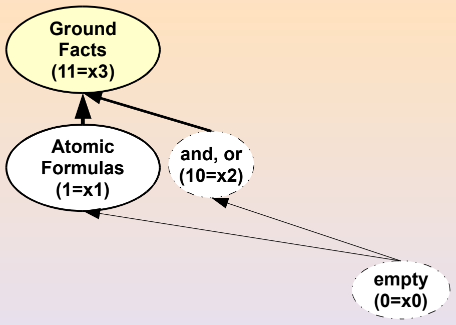


The Relax NG modularization is more fine-grained than the XSD modularization. For example, the Atomic Formulas modules together with common modules defining the root element <RuleML> and performatives, such as <Assert>, provides a minimal language that has the lowest expressive power of the sublanguages available from the RuleML MYNG GUI.

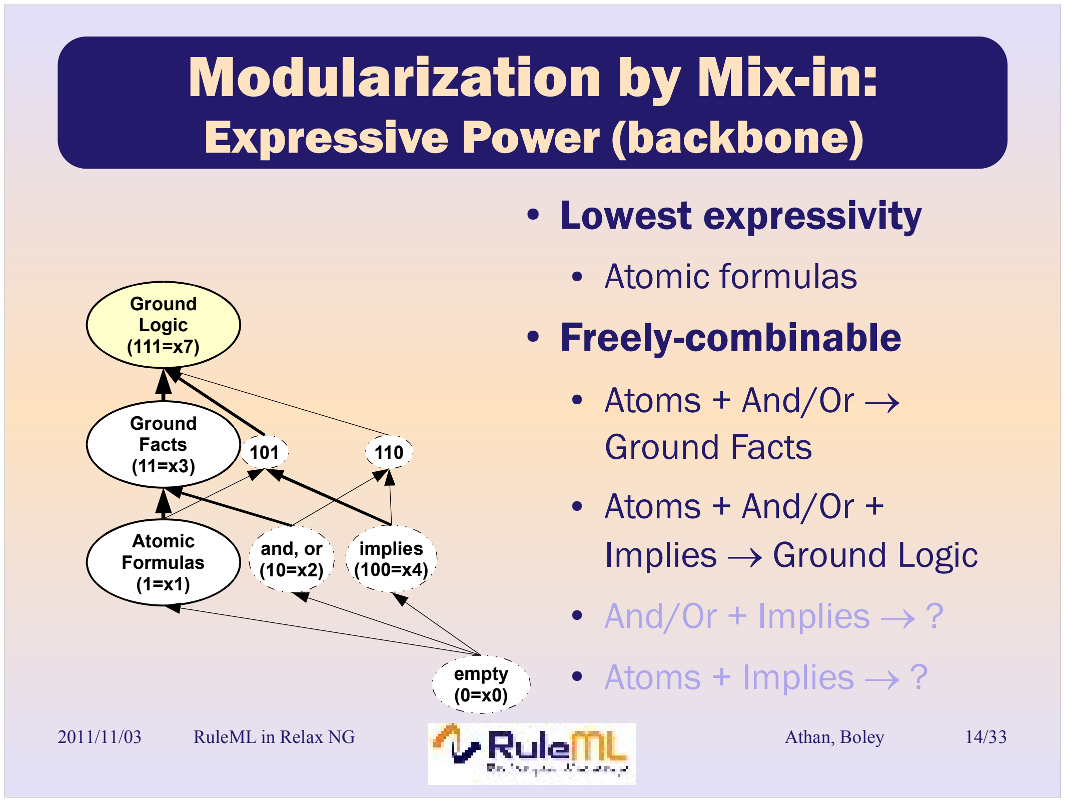


Modularization by Mix-in: Expressive Power (backbone)

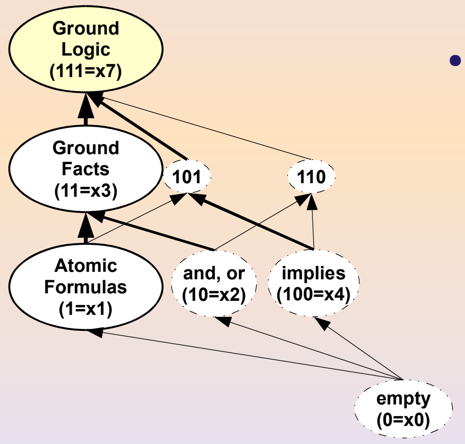
- **Lowest expressivity**
  - Atomic formulas
- **Freely-combinable**
  - Atoms + And/Or → Ground Facts
  - Atoms + And/Or + Implies → Ground Logic
  - And/Or + Implies → ?
  - Atoms + Implies → ?



Mixing-in the And/Or module as well as Atomic Formulas produces the Ground Fact level of expressive power, with conjunction and disjunction of atomic formulas.

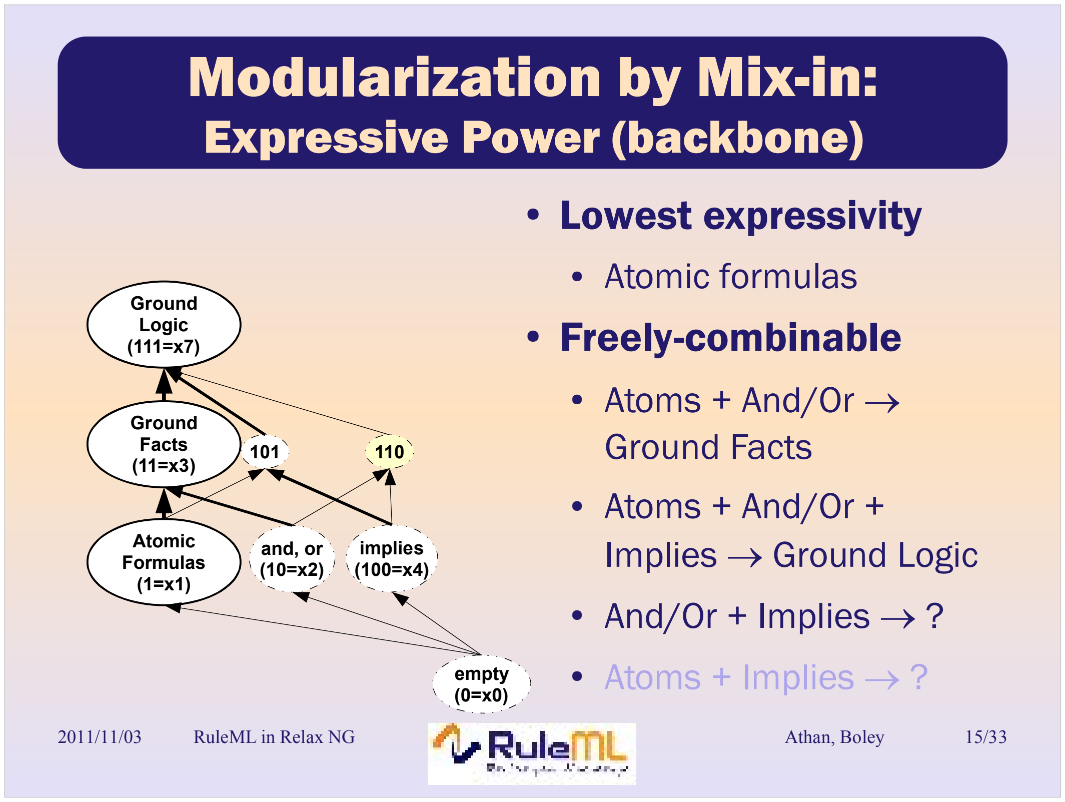


Modularization by Mix-in: Expressive Power (backbone)

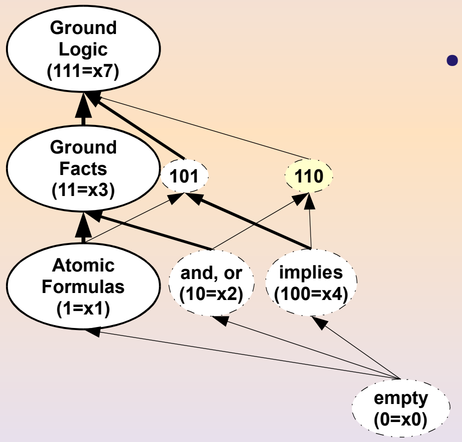


- **Lowest expressivity**
  - Atomic formulas
- **Freely-combinable**
  - Atoms + And/Or → Ground Facts
  - Atoms + And/Or + Implies → Ground Logic
  - And/Or + Implies → ?
  - Atoms + Implies → ?

Mixing-in the Implies module with And/Or and Atom gives the Ground Logic level of expressive power.



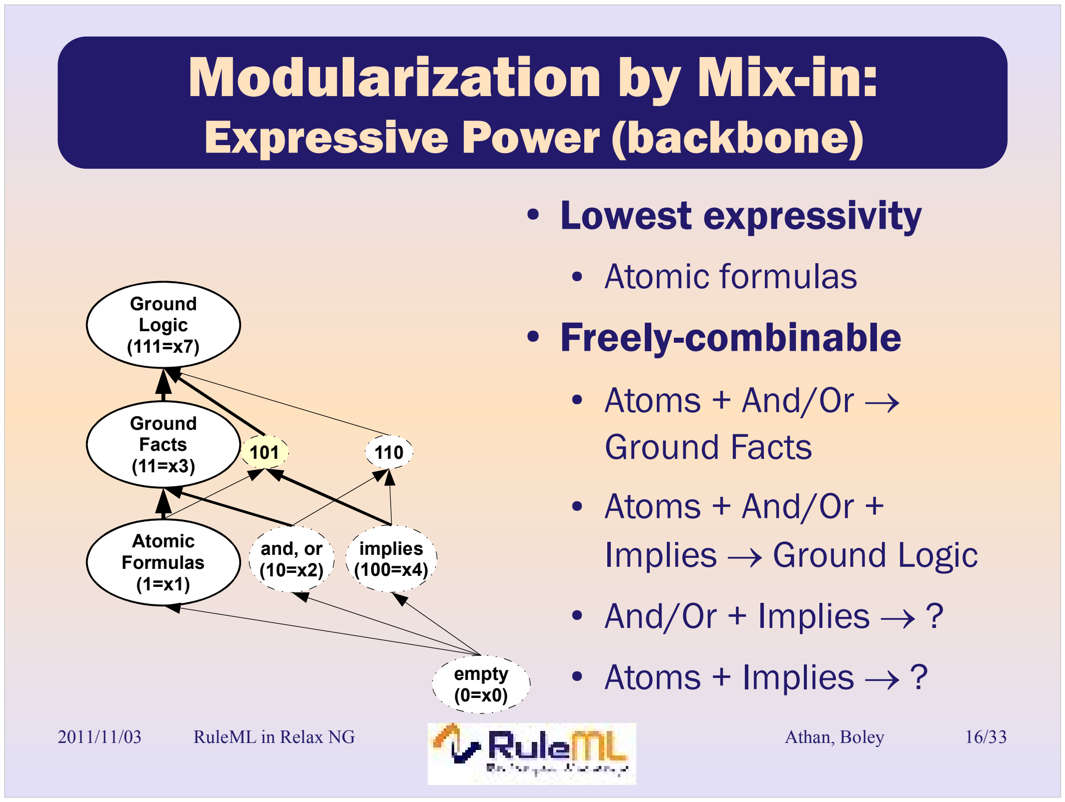
Modularization by Mix-in: Expressive Power (backbone)



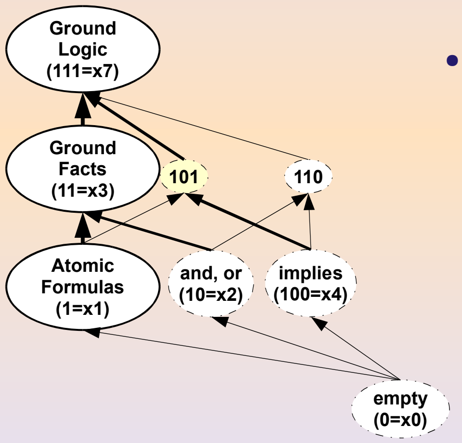
- **Lowest expressivity**
  - Atomic formulas
- **Freely-combinable**
  - Atoms + And/Or → Ground Facts
  - Atoms + And/Or + Implies → Ground Logic
  - And/Or + Implies → ?
  - Atoms + Implies → ?

What happens if we combine the modules for And/Or and Implies, but not Atomic Formulas?

It is a non-empty language, because the empty <And/> and <Or/> are terminal symbols, representing Truth and Falsity, respectively, but has no vocabulary.



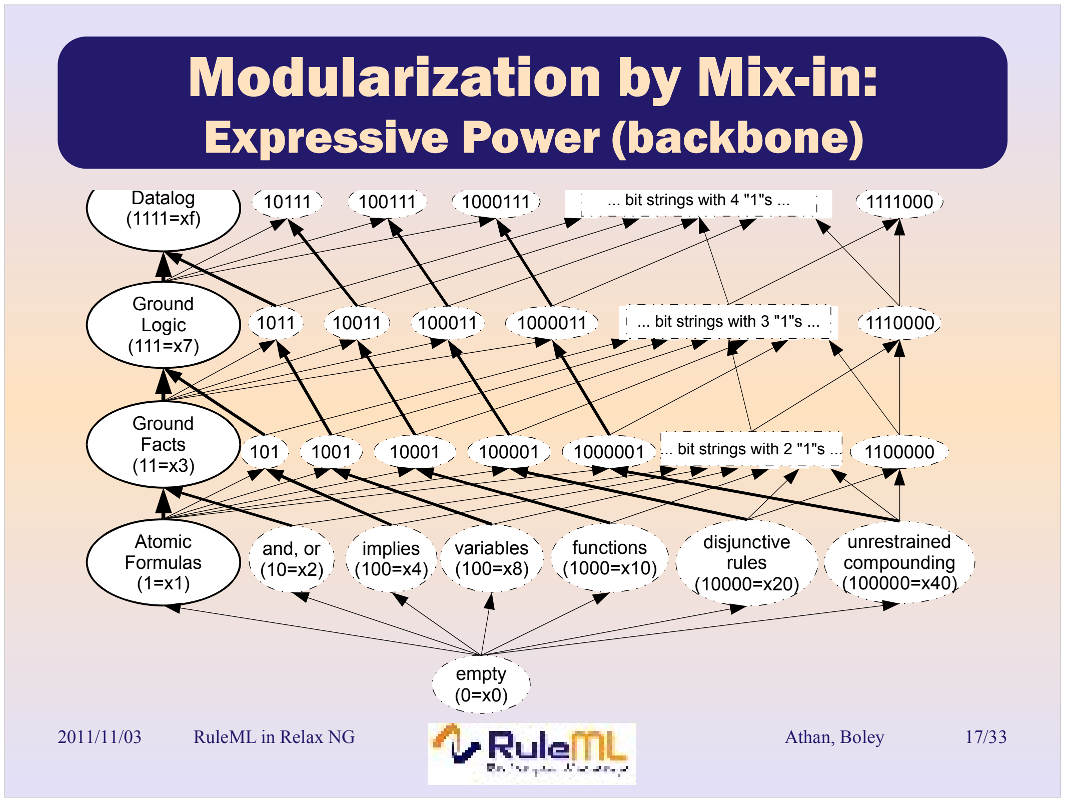
Modularization by Mix-in: Expressive Power (backbone)



- **Lowest expressivity**
  - Atomic formulas
- **Freely-combinable**
  - Atoms + And/Or → Ground Facts
  - Atoms + And/Or + Implies → Ground Logic
  - And/Or + Implies → ?
  - Atoms + Implies → ?

What happens if we combine the modules for Atomic Formulas and Implies?

This is a viable language, allowing compound implications of Atomic formulas but without conjunction and disjunction.



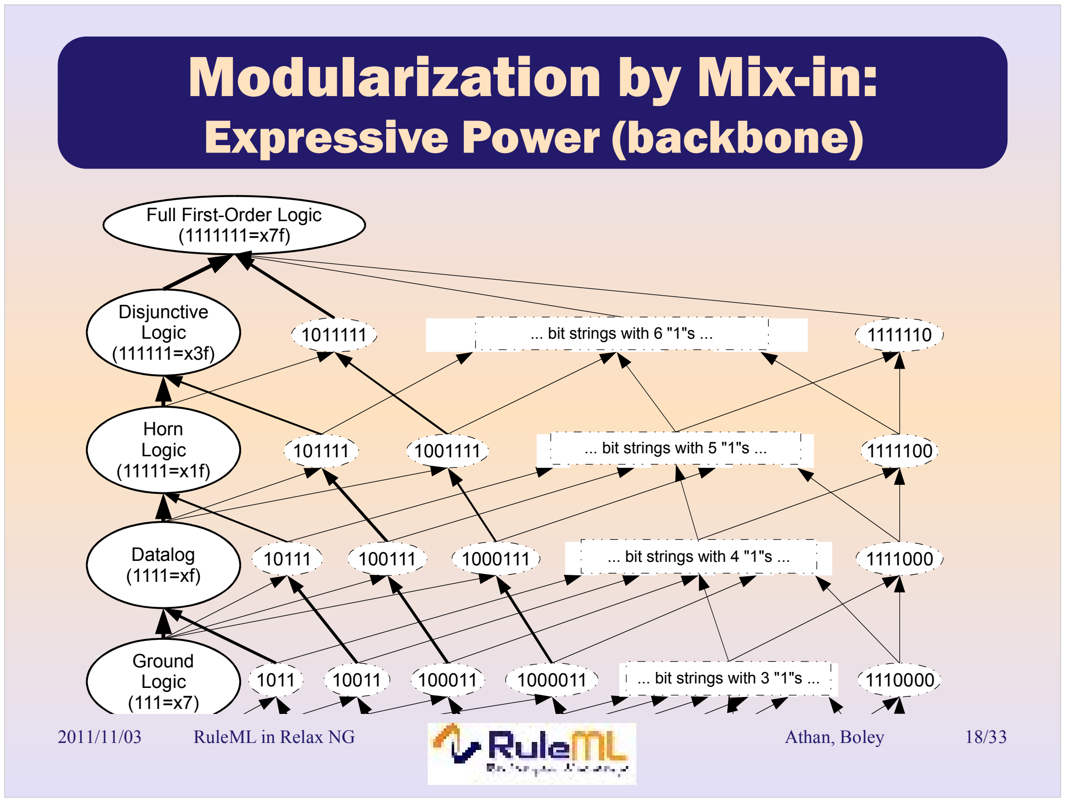
Modularization by Mix-in: Expressive Power (backbone)

The full lattice of all sublanguages that could be created by freely combining these modules would be very large.

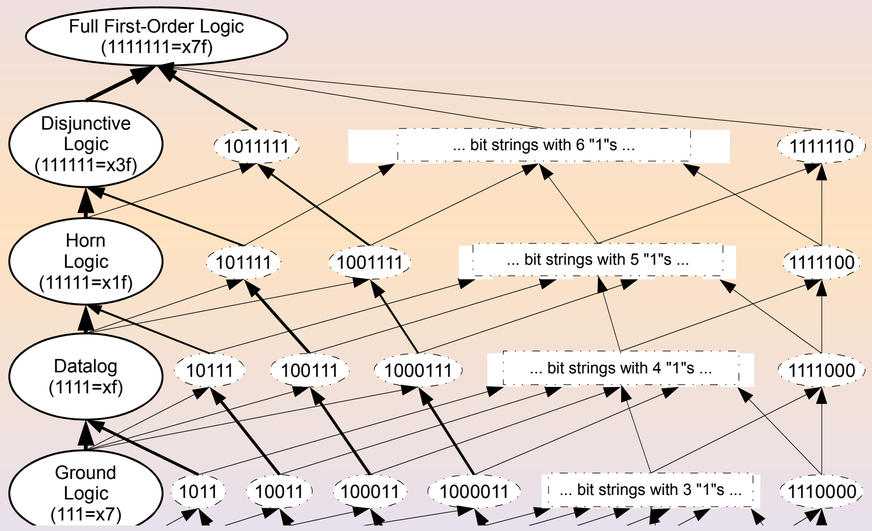
Only the options on the left are available from the RuleML MYNG GUI.

For advanced users, the other options are available from the parameterized schema by direct manipulation of the query string – or by directly modifying a local copy of the schema driver.

A total of 7 levels of expressive power are available from the GUI, from Atomic Formulas Only

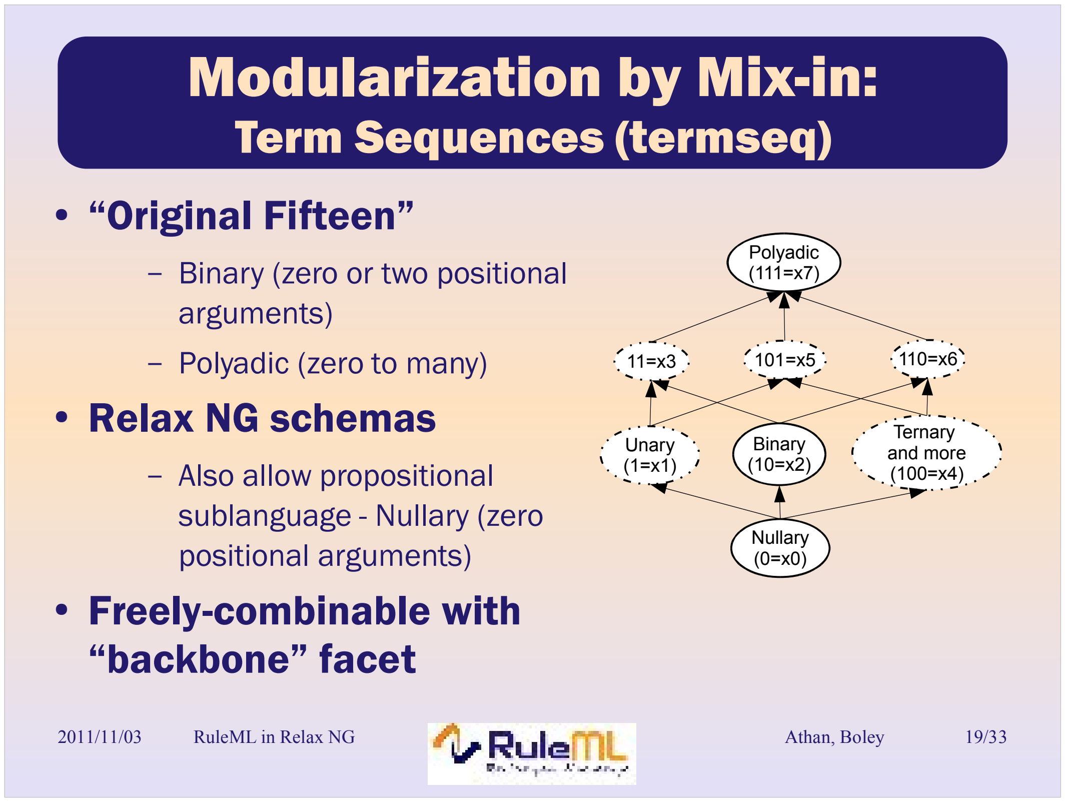


Modularization by Mix-in: Expressive Power (backbone)



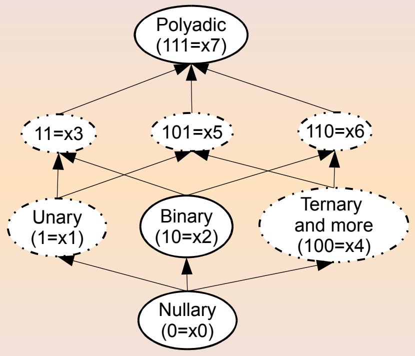
Full First-Order Logic (1111111=x7f)
Disjunctive Logic (111111=x3f)
1011111
... bit strings with 6 "1"s ...
1111110
Horn Logic (11111=x1f)
101111
1001111
... bit strings with 5 "1"s ...
1111100
Datalog (1111=xf)
10111
100111
1000111
... bit strings with 4 "1"s ...
1111000
Ground Logic (111=x7)
1011
10011
100011
1000011
... bit strings with 3 "1"s ...
1110000

... to Full First-Order Logic.



Modularization by Mix-in: Term Sequences (termseq)

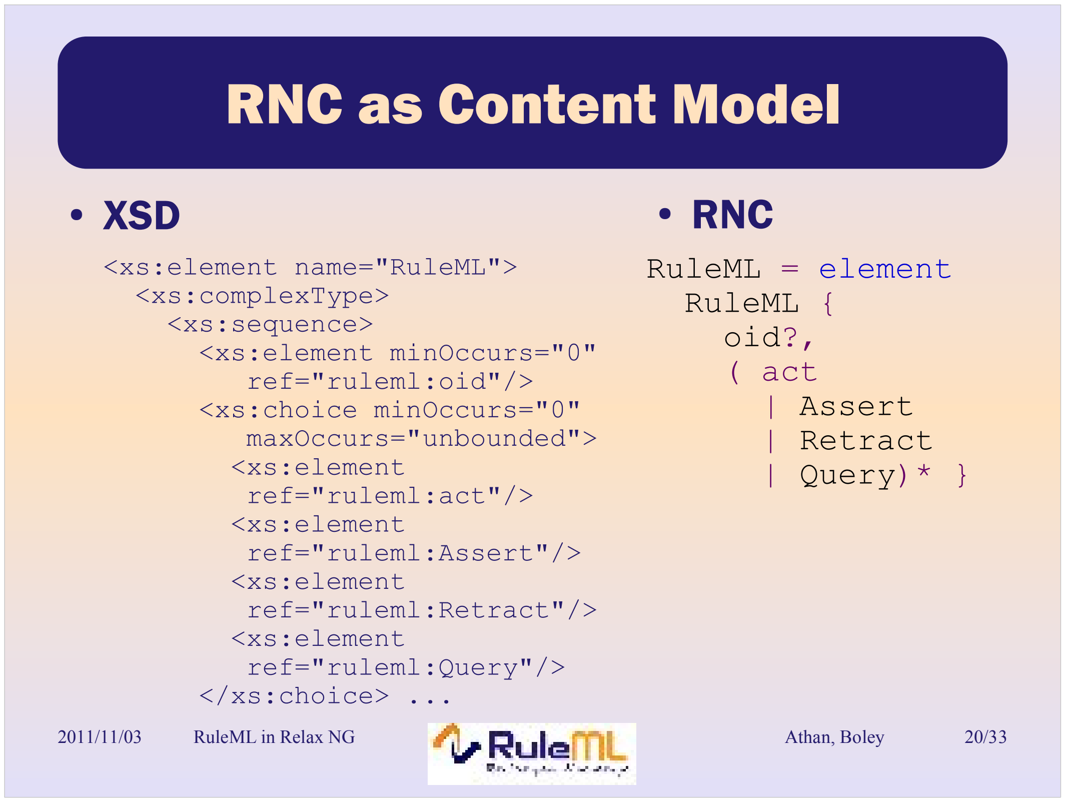
- **"Original Fifteen"**
  - Binary (zero or two positional arguments)
  - Polyadic (zero to many)
- **Relax NG schemas**
  - Also allow propositional sublanguage - Nullary (zero positional arguments)
- **Freely-combinable with "backbone" facet**



The original fifteen RuleML sublanguages allowed either binary or polyadic positional arguments.

The Relax NG schemas also allow no positional arguments (nullary atomic formulas), corresponding to a propositional sublanguage. This option could also be used for a pure frame language (slots only).

The lattice encoding is based on separate vertices for Unary and Ternary+ sequences. Although these are not implemented at this time, such modules may be needed in the future, e.g. for the implementation of SWSL.



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**

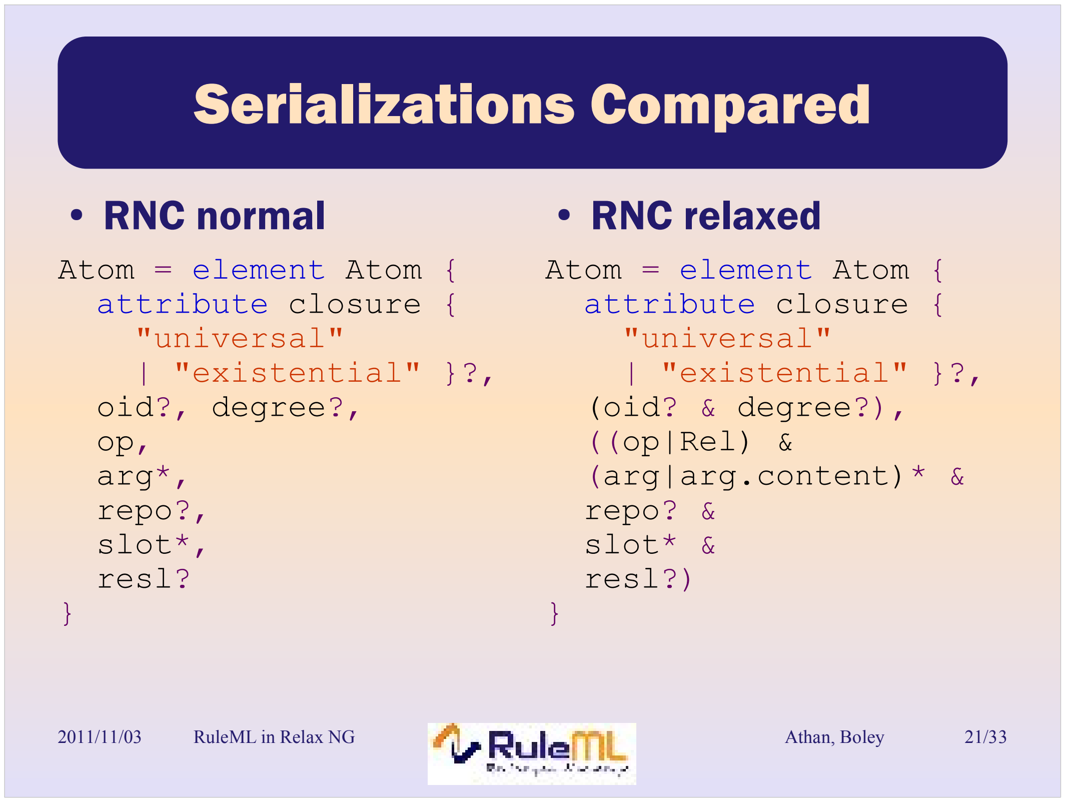
```
RuleML = element
  RuleML {
    oid?,
    ( act
      | Assert
      | Retract
      | Query)* }
```

2011/11/03 RuleML in Relax NG Athan, Boley 20/33

This slide shows a side-by-side view of an XSD and the corresponding RNC schema as a "content model". RNC syntax should look very familiar to DTD users. However, RNC is not a step backwards. RNC is more expressive than XSD which is in turm more expressive than DTD.

The RNC schema defines a pattern, call "RuleML" which matches an element with name "RuleML" having content model of an optional oid element and zero to many of a choice of act, Assert, Retract or Query elements.

This example illustrates striping, with capitalized Node stripes (RuleML, Assert, ...) alternatiing with lower-case edge stripes (act). If redundant, stripes may be skipped.



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?)
}
```

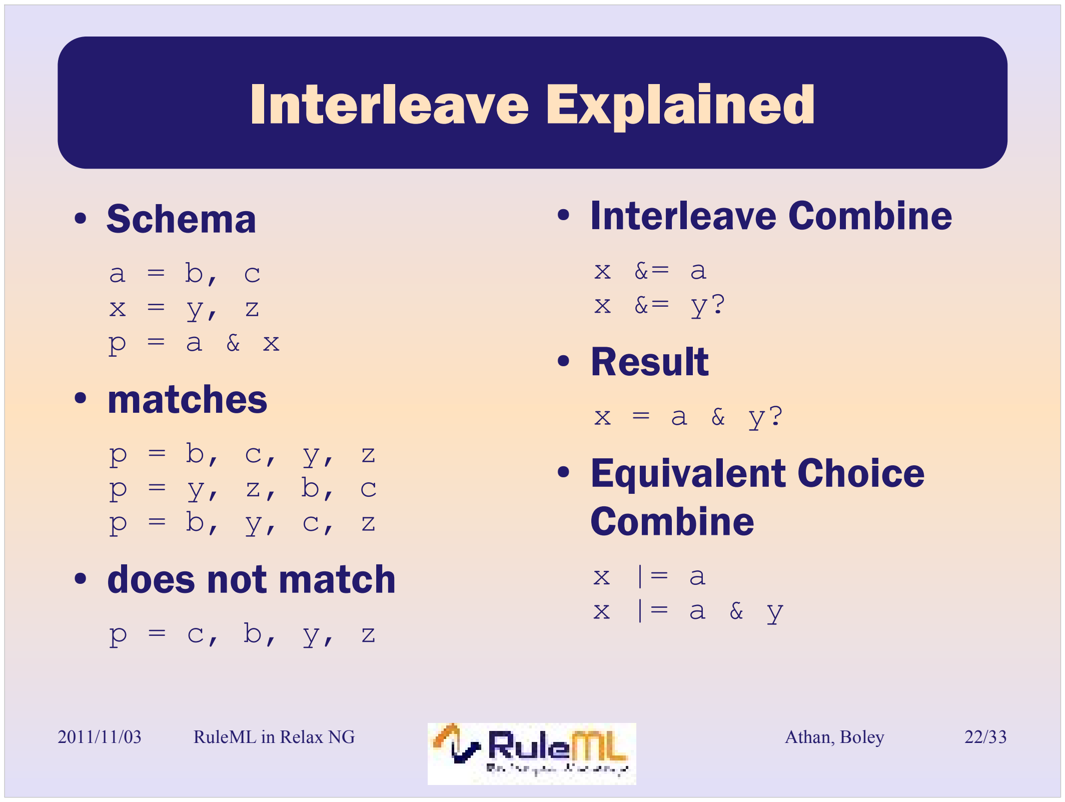
This slide shows a side-by-side view of monolithic RNC schemas automatically generated, for the most part, by Trang from the modular schemas for naffologeq.

Both have an optional closure attribute which can take on two possible values, as well as optional oid, degree, required op, zero to many arg and/or slots and optional repo and resl.

The normal serialization uses commas, indicating a position-dependent sequence.

The relaxed serialization uses ampersands &, the interleave symbol in RNC. This indicates a position-independent interleaved sequence

The relaxed content model also shows stripe-skipping for op and arg.



Interleave Explained

- **Schema**

```
a = b, c
x = y, z
p = a & x
```

- **matches**

```
p = b, c, y, z
p = y, z, b, c
p = b, y, c, z
```

- **does not match**

```
p = c, b, y, z
```

- **Interleave Combine**

```
x &= a
x &= y?
```

- **Result**

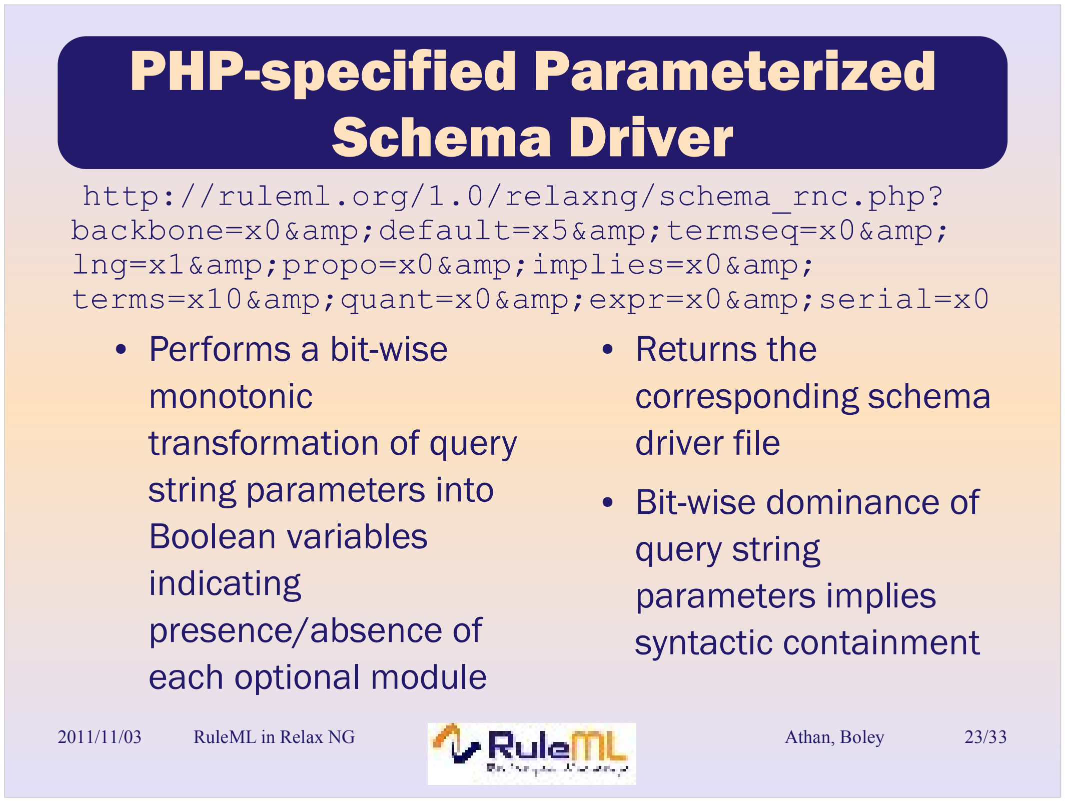
```
x = a & y?
```

- **Equivalent Choice Combine**

```
x |= a
x |= a & y
```

The RNC interleave operator is one aspect of the grammar that was not available in DTDs. We illustrate the meaning of this operator with a simple example. Consider a schema with three pattern definitions as shown here, for sequence patterns a and x, and p defined as the interleaving of sequences a and x. Then the pattern p will match a sequence which is the concatenation of a and x. It will also match the concatenation of x and a. Further it will match a sequence where the elements of a and x are intermingled, provided the relative order of the original sequences is honored.

The interleave combine is a method for merging pattern definitions through an interleave operation. As long as the extension pattern is optional, then the modules can be reformulated without the interleave combine.


PHP-specified Parameterized Schema Driver

```
http://ruleml.org/1.0/relaxng/schema_rnc.php?
backbone=x0&amp;default=x5&amp;termseq=x0&amp;
lng=x1&amp;propo=x0&amp;implies=x0&amp;
terms=x10&amp;quant=x0&amp;expr=x0&amp;serial=x0
```

- Performs a bit-wise monotonic transformation of query string parameters into Boolean variables indicating presence/absence of each optional module
- Returns the corresponding schema driver file
- Bit-wise dominance of query string parameters implies syntactic containment

The other major component of MYNG is the parameterized schema driver, implemented as a PHP script accessed through a URL with query string.

The passed parameter values are transformed by a "bit-wise monotic" transformation (that is, if two sets of parameters are bit-wise comparable, then the transformed values are also comparable, with the same order.) The results of the transformation signifies presence or absence of individual modules. The output of the PHP script is the schema driver file with include statements for each modules with a positive bit value. Therefore, bit-wise dominance of the query string parameters indicates grammar containment of the corresponding schemas.

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

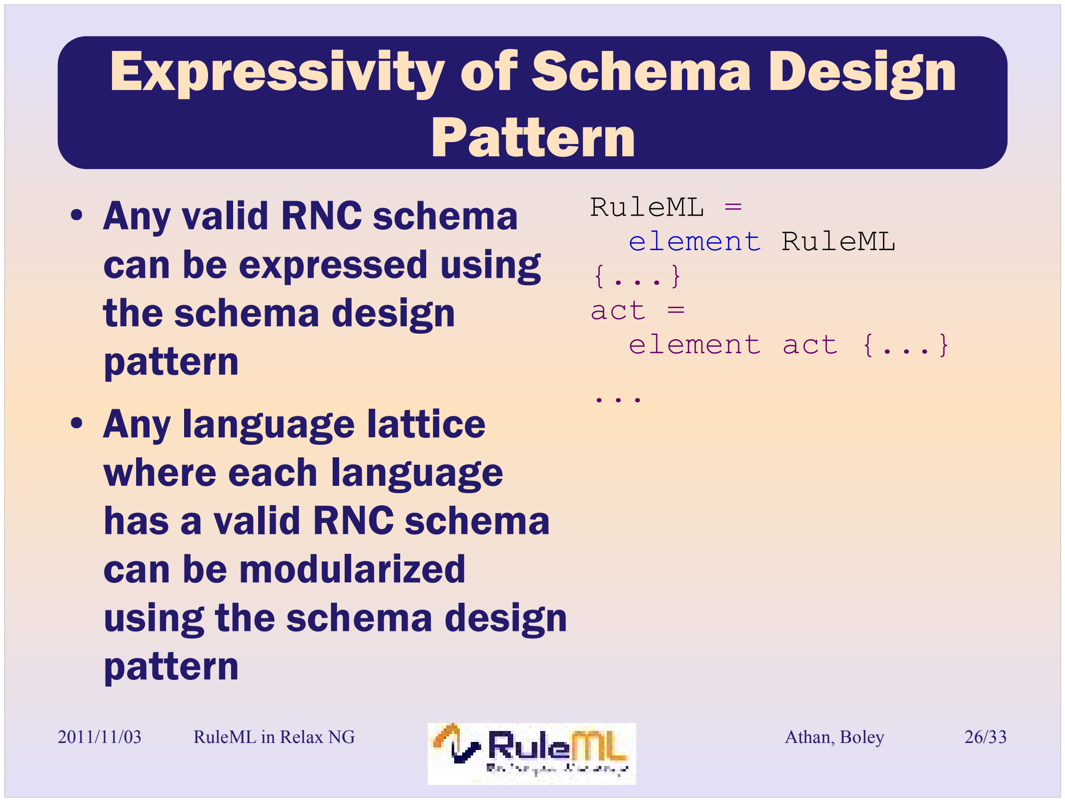
- **Segregated Names**
  - Choice combine
  - No combine
  - Interleave combine
    - &= empty
    - &= ...?
    - &= ...*
- **Joins by union, not redefinition**

```
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 {...}
...
```

2011/11/03 RuleML in Relax NG Athan, Boley 26/33

The results stated here are trivial to prove.

The simplified form of any valid RNC schema has only no-combine definitions, and those will always match the schema design pattern.

One modularization of a language lattice is the set of all stand-alone schemas for the languages in the lattice, except for changes of names of patterns to avoid collisions. These "modules" may be combined because Relax NG allows ambiguous patterns.

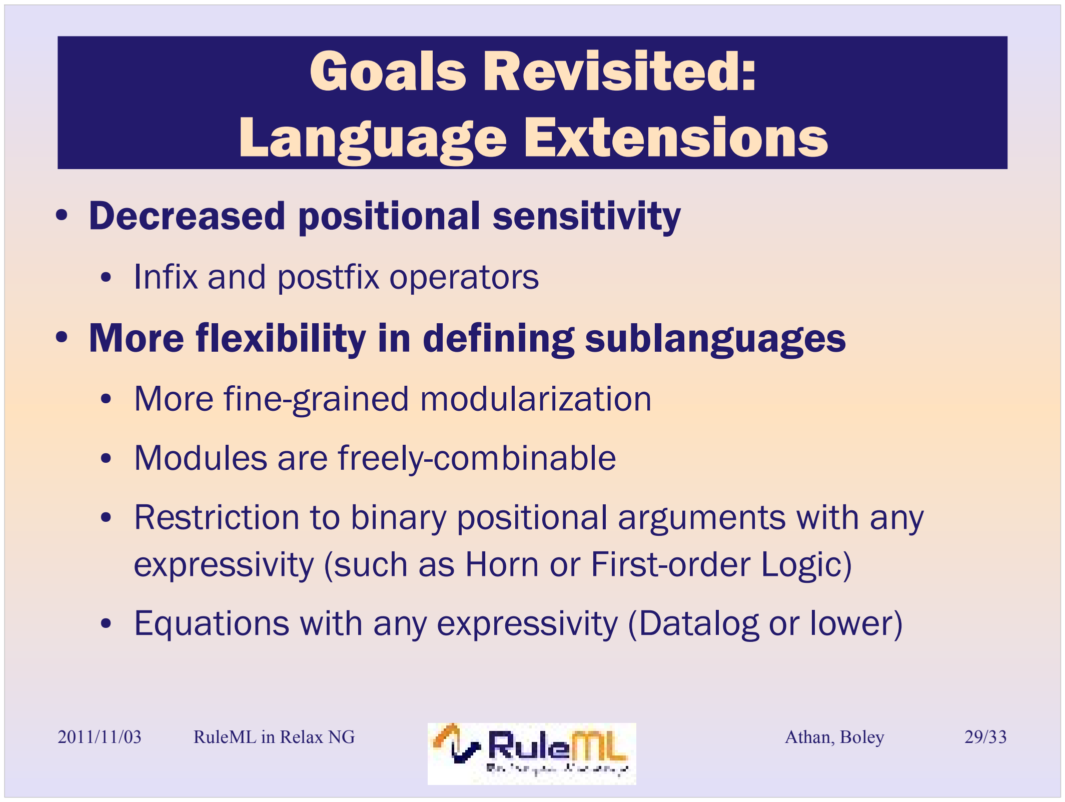
While not an efficient modularization, it does satisfy the schema design pattern. It could serve as the starting point of an optimization algorithm to generate a more parsimonious modularization.

Status of Re-engineering

Task	Version 0.91	Version 1.0
Hand-written XSD Schemas Patched	✓	✓
Relax NG Modules	✓	✓
MYNG: PHP-specified Parametrized Schema Driver	✓	✓
MYNG: GUI	✓	✓
On-the-fly Zip Archives	✓	✓
Upgrader XSLT		✓
Normalizer XSLT		In progress

Status of Re-engineering, cont.

Task	Version 0.91	Version 1.0
Auto-generated XSDs for Normal Serialization	✓	✓
Meta-schemas for Base and Expansion Modules	✓	✓
HTML documentation	✓	✓
XSD Content-model document (pdf)		✓
Statistically-random instance test suite	✓	
Simplified RNC (normal and relaxed)	✓	✓



Goals Revisited: Language Extensions

- **Decreased positional sensitivity**
  - Infix and postfix operators
- **More flexibility in defining sublanguages**
  - More fine-grained modularization
  - Modules are freely-combinable
  - Restriction to binary positional arguments with any expressivity (such as Horn or First-order Logic)
  - Equations with any expressivity (Datalog or lower)

RuleML is a family of languages for Web rule interchange that was originally specified in Document Type Definitions (DTDs), then switched to XML Schema Definition Language (XSD) schemas.

Here we present a re-engineering and re-conceptualization of the non-SWSL portion of the Derivation Rules subfamily of RuleML in the Relax NG Compact (RNC) schema syntax.

The goals of this effort are

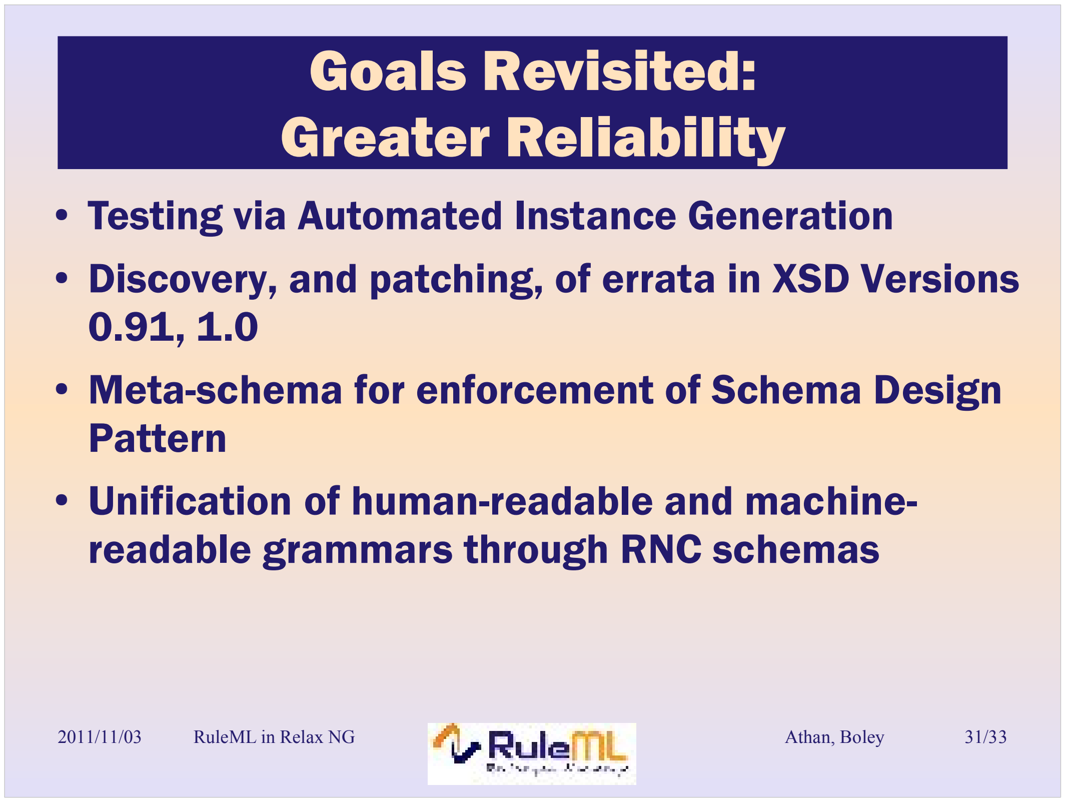
* RuleML language extensions, including decreased positional sensitivity and greater flexibility in defining sublanguages

* greater reliability.

* Greater automation via the use of web applications (preferred) or desktop software.

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.**


Goals Revisited: Greater Reliability

- **Testing via Automated Instance Generation**
- **Discovery, and patching, of errata in XSD Versions 0.91, 1.0**
- **Meta-schema for enforcement of Schema Design Pattern**
- **Unification of human-readable and machine-readable grammars through RNC schemas**

RuleML is a family of languages for Web rule interchange that was originally specified in Document Type Definitions (DTDs), then switched to XML Schema Definition Language (XSD) schemas.

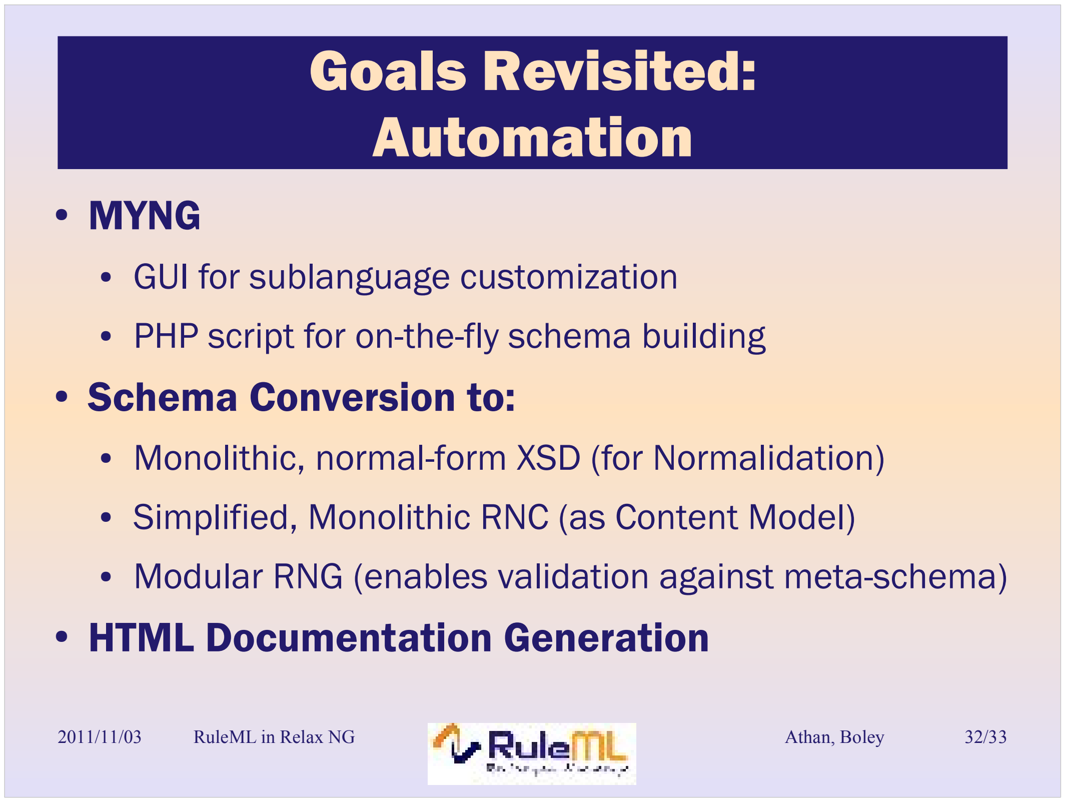
Here we present a re-engineering and re-conceptualization of the non-SWSL portion of the Derivation Rules subfamily of RuleML in the Relax NG Compact (RNC) schema syntax.

The goals of this effort are

* RuleML language extensions, including decreased positional sensitivity and greater flexibility in defining sublanguages

* greater reliability.

* Greater automation via the use of web applications (preferred) or desktop software.



Goals Revisited: Automation

- **MYNG**
  - GUI for sublanguage customization
  - PHP script for on-the-fly schema building
- **Schema Conversion to:**
  - Monolithic, normal-form XSD (for Normalidation)
  - Simplified, Monolithic RNC (as Content Model)
  - Modular RNG (enables validation against meta-schema)
- **HTML Documentation Generation**

RuleML is a family of languages for Web rule interchange that was originally specified in Document Type Definitions (DTDs), then switched to XML Schema Definition Language (XSD) schemas.

Here we present a re-engineering and re-conceptualization of the non-SWSL portion of the Derivation Rules subfamily of RuleML in the Relax NG Compact (RNC) schema syntax.

The goals of this effort are

* RuleML language extensions, including decreased positional sensitivity and greater flexibility in defining sublanguages
* greater reliability.
* Greater automation via the use of web applications (preferred) or desktop software.

Future Developments

- **Version 1.0**
  - Normalizing XSLT
  - From Feedback
    - Improved Documentation
    - Use cases
    - Improved MYNG Usability

- **Version 1.1**
  - Focus on alignment with semantics
  - Complete implementation of Fuzzy RuleML
  - Separate Query sublanguage
  - User-extensibility (beyond customization)