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

Relax NG Schema Language: Features

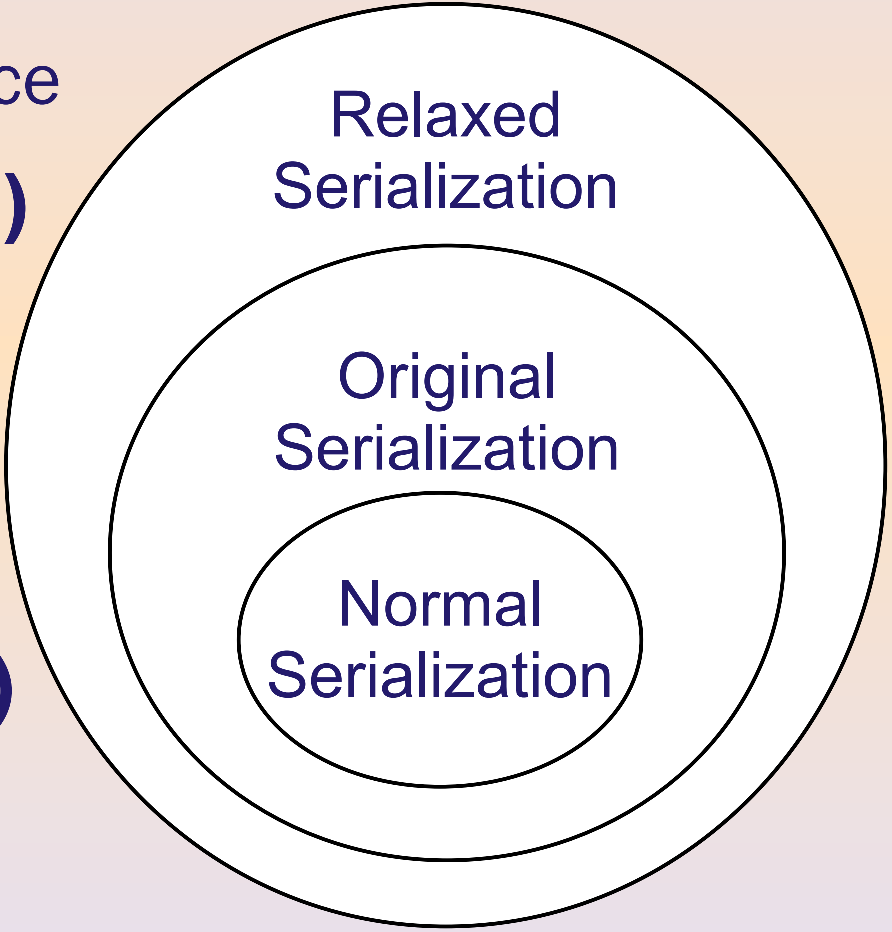
- **Decreased Positional Sensitivity**
  - Sequence interleave
- **Greater Flexibility in Modularization**
  - Combining definitions: choice and interleave
- **Closure under Union and Intersection**
- **More Expressive than XSD and DTD**
- **Compact Syntax (RNC)**
  - Unification of Human-Readable and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

•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

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



Decreased Positional Sensitivity: Example of Relaxed Serialization

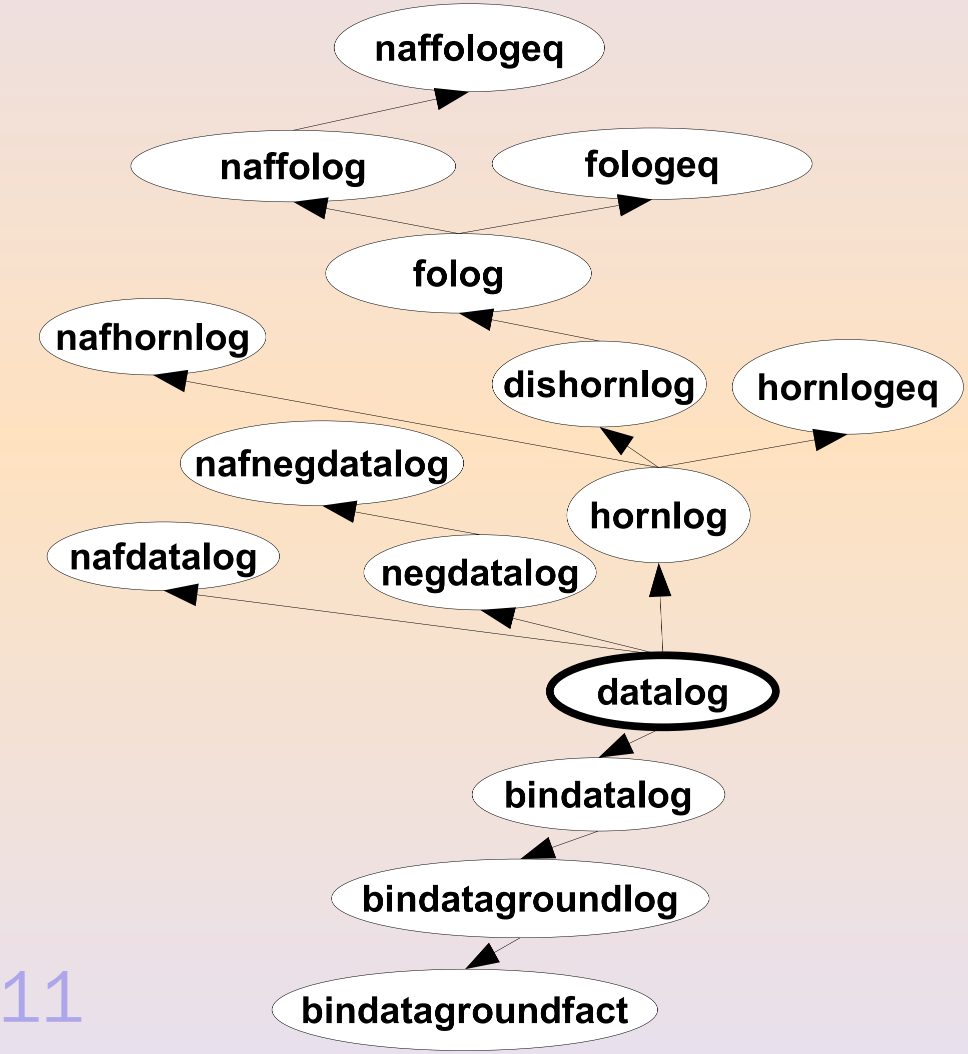
```
<Atom>
  <arg index=”1”>
    <Var>customer</Var>
  </arg>

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

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

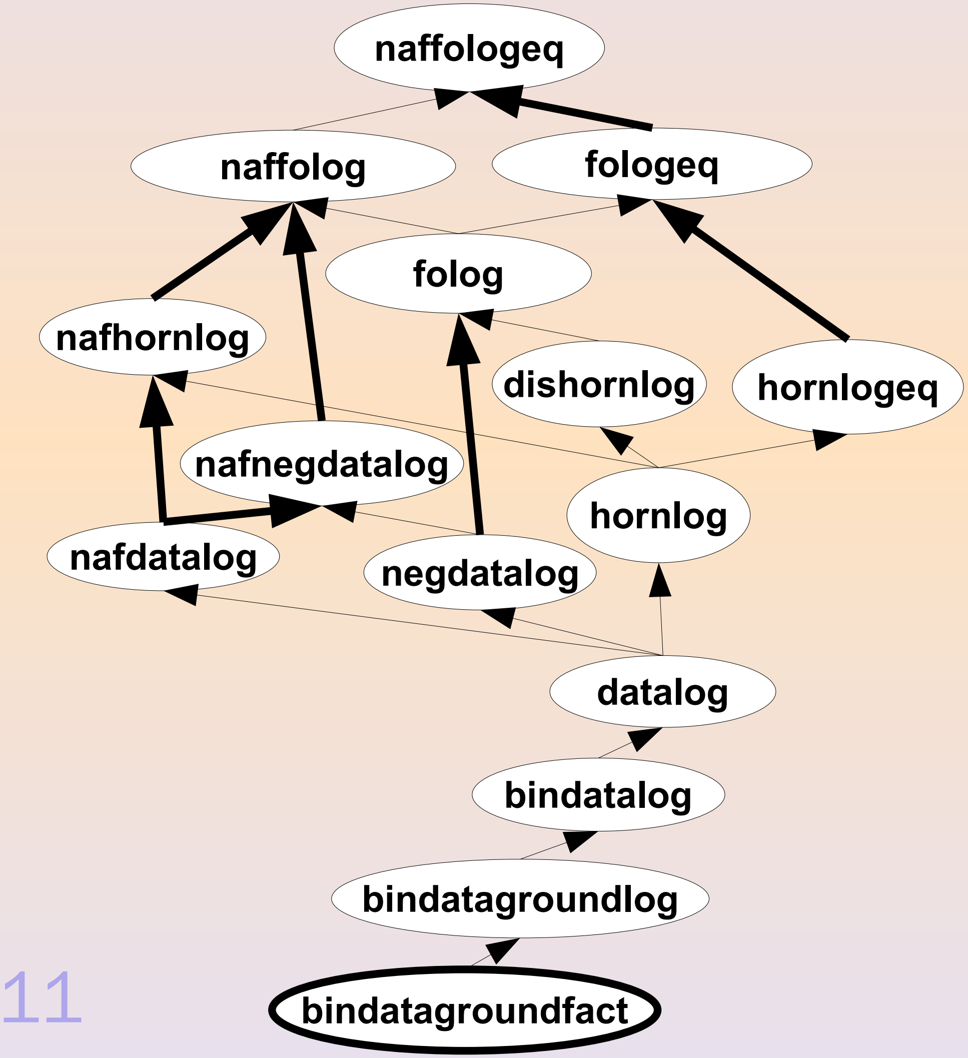
Modularization: "Original Fifteen" (non-SWSL)

- 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



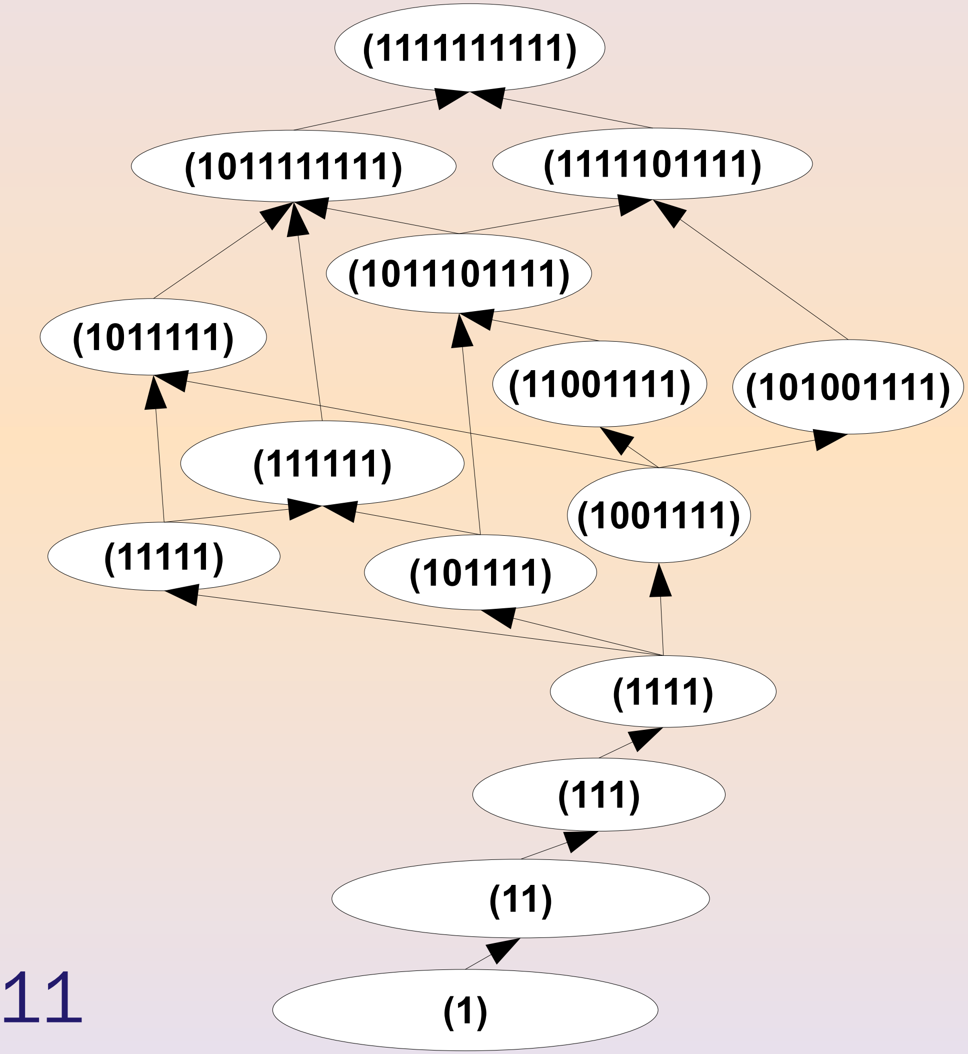
Modularization:
Original Fifteen FOL

- 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

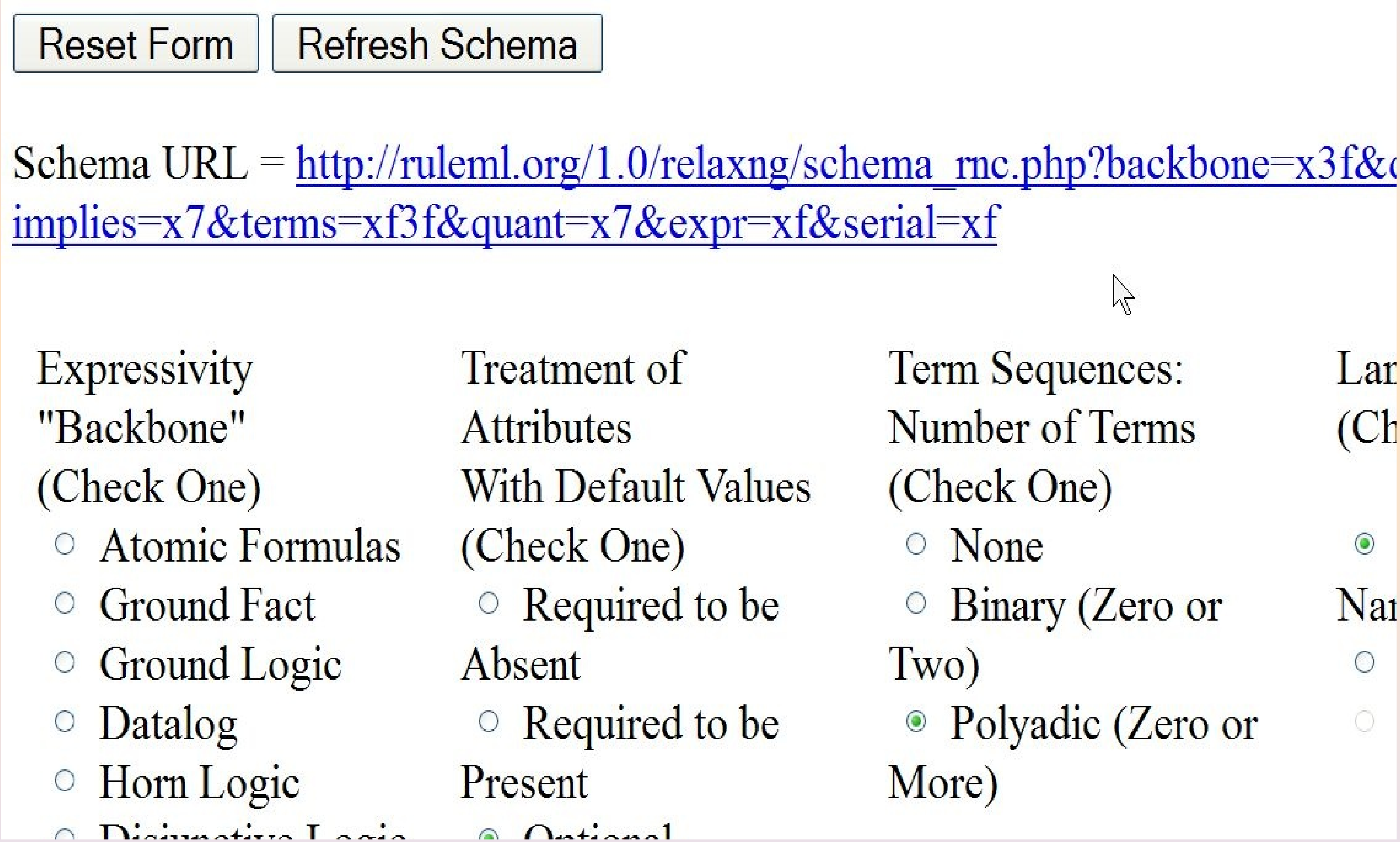


Modularization:
Original Fifteen FOL

- **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 http://ruleml.org/1.0/myng/



Schema URL = http://ruleml.org/1.0/relaxng/schema_rnc.php?backbone=x3f&c
implies=x7&terms=xf3f&quant=x7&expr=xf&serial=xf

Expressivity "Backbone" (Check One)

- ○ Atomic Formulas
- ○ Ground Fact
- ○ Ground Logic
- ○ Datalog
- ○ Horn Logic
- ○ Disjunctive Logic

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)

Lar
(Ch

- ●

Nar

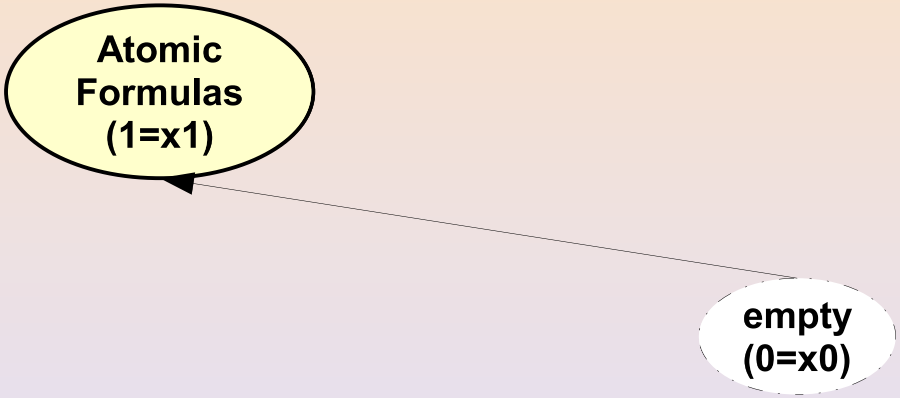
- ○
- ○

On-the-Fly Instance Validation

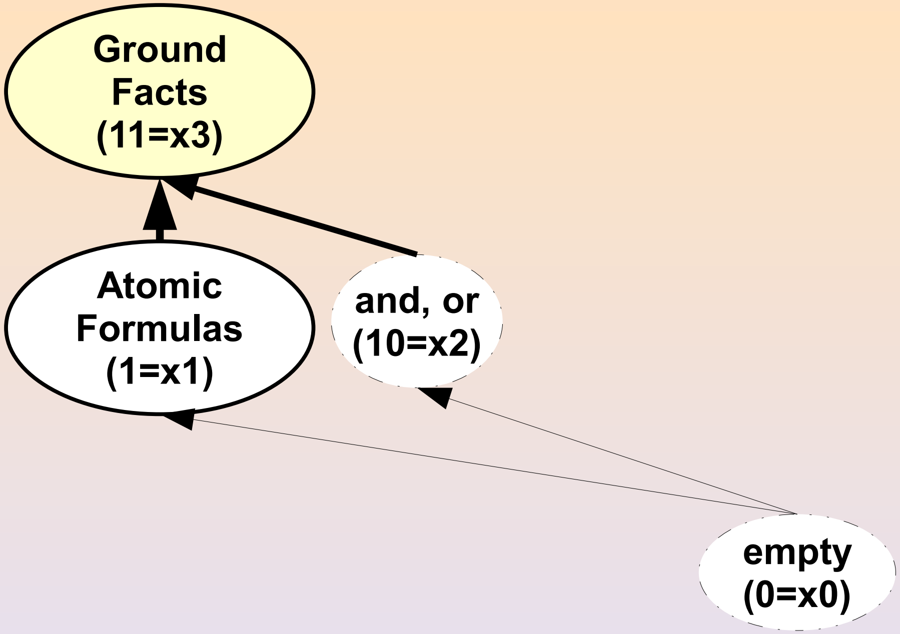
```
<?xml-model href="http://ruleml.org/1.0/
  relaxng/schema_rnc.php?
  backbone=x0&amp;terms=x10&amp;..."
  type="application/relax-ng-compact-syntax"?>
<RuleMLxmlns="...">
  <Assert>  <formula>
    <Equal>
      <left><Ind>Lady Gaga</Ind></left>
      <right><Ind>Stefani Joanne Angelina
         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

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 → ?

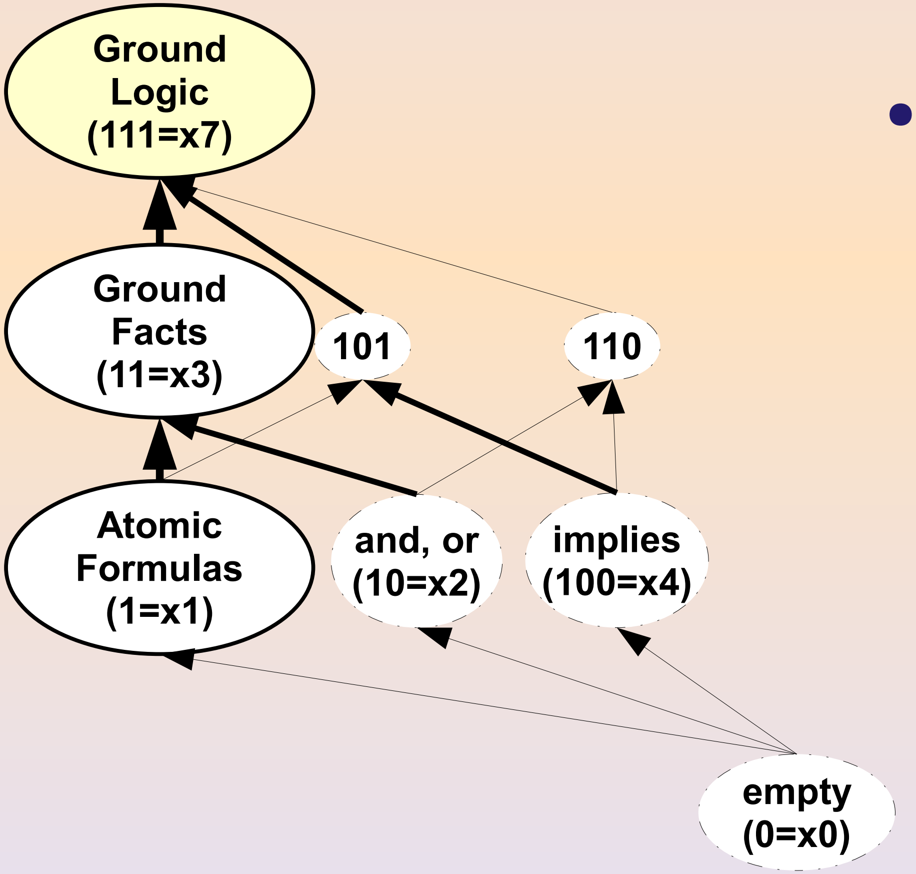


Modularization by Mix-in:
Expressive Power (backbone)



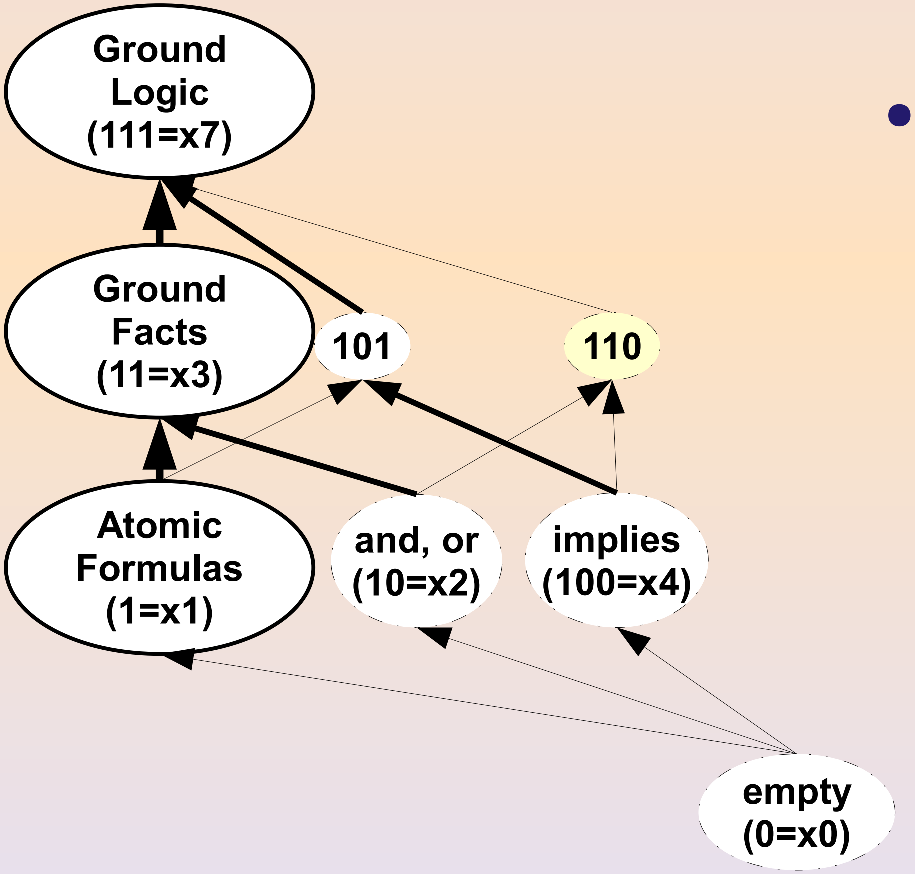
- **Lowest expressivity**
  - Atomic formulas
- **Freely-combinable**
  - Atoms + And/Or → Ground Facts
  - Atoms + And/Or + Implies → Ground Logic
  - And/Or + Implies → ?
  - Atoms + Implies → ?

Modularization by Mix-in: Expressive Power (backbone)



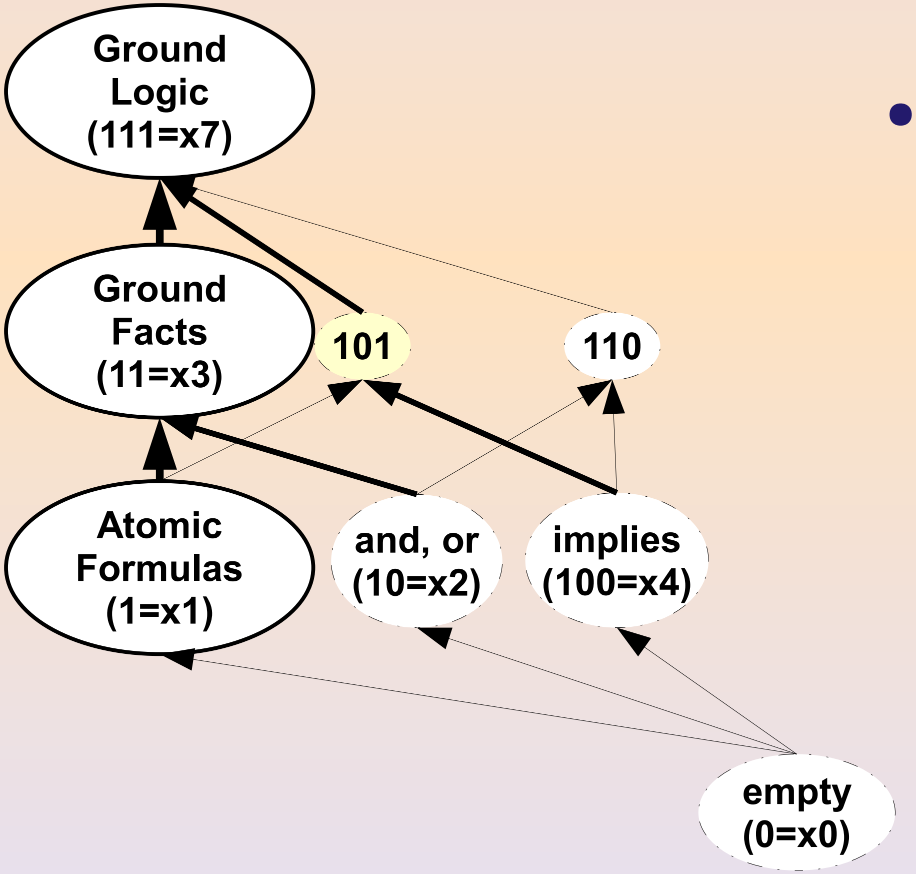
- **Lowest expressivity**
  - Atomic formulas
- **Freely-combinable**
  - Atoms + And/Or → Ground Facts
  - Atoms + And/Or + Implies → Ground Logic
  - And/Or + Implies → ?
  - Atoms + Implies → ?

Modularization by Mix-in:
Expressive Power (backbone)



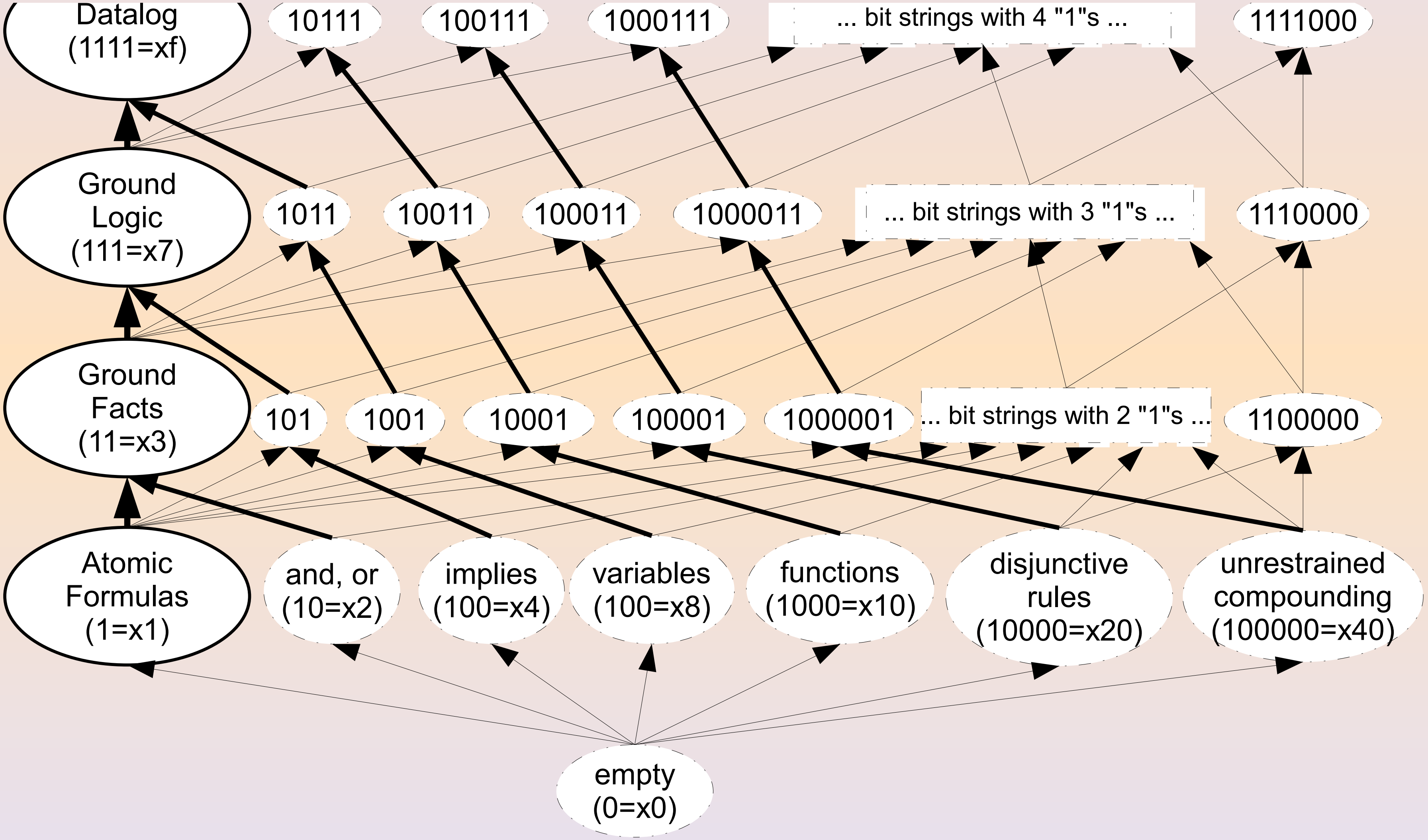
- **Lowest expressivity**
  - Atomic formulas
- **Freely-combinable**
  - Atoms + And/Or → Ground Facts
  - Atoms + And/Or + Implies → Ground Logic
  - And/Or + Implies → ?
  - Atoms + Implies → ?

Modularization by Mix-in:
Expressive Power (backbone)

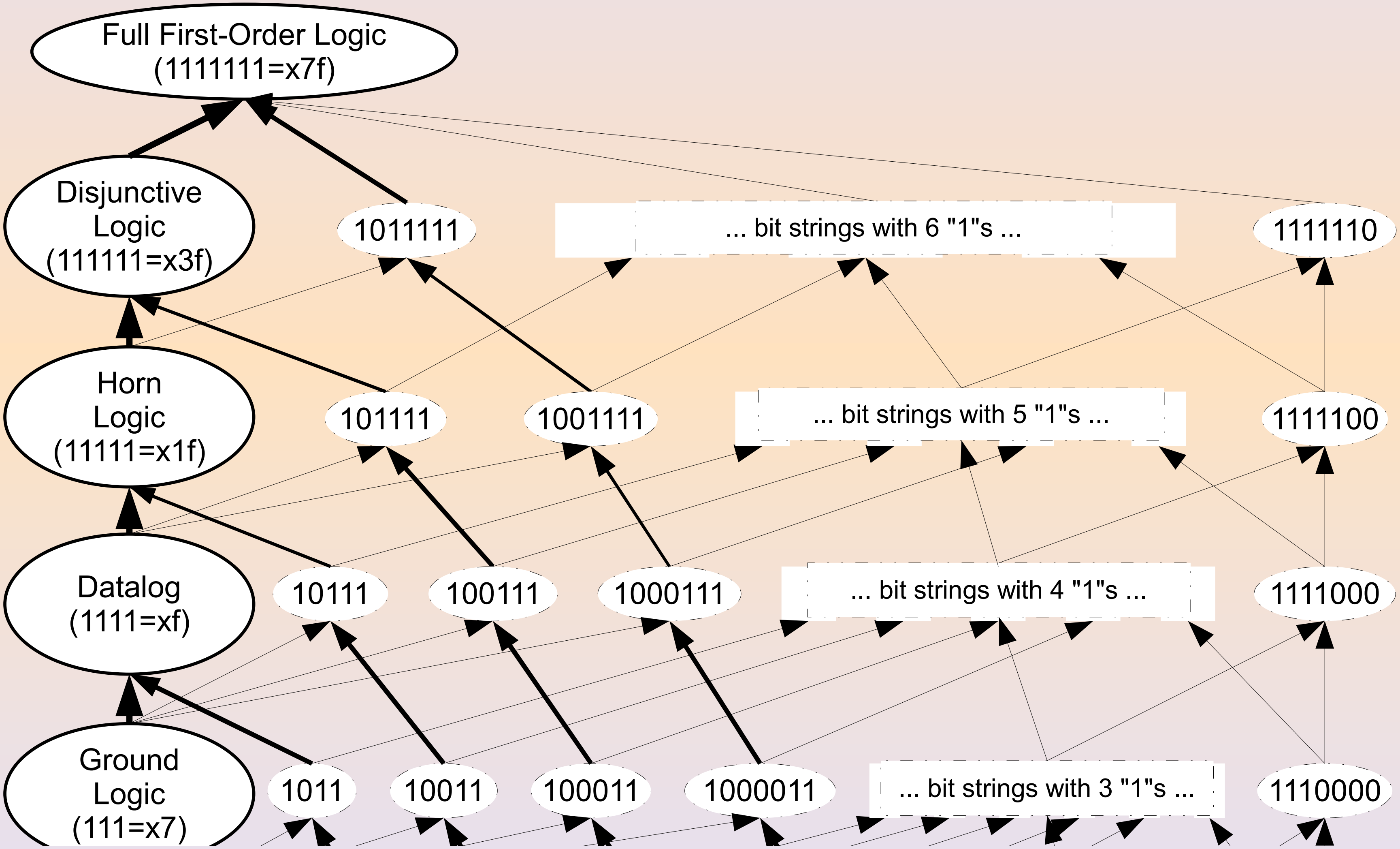


- **Lowest expressivity**
  - Atomic formulas
- **Freely-combinable**
  - Atoms + And/Or → Ground Facts
  - Atoms + And/Or + Implies → Ground Logic
  - And/Or + Implies → ?
  - Atoms + Implies → ?

Modularization by Mix-in: Expressive Power (backbone)

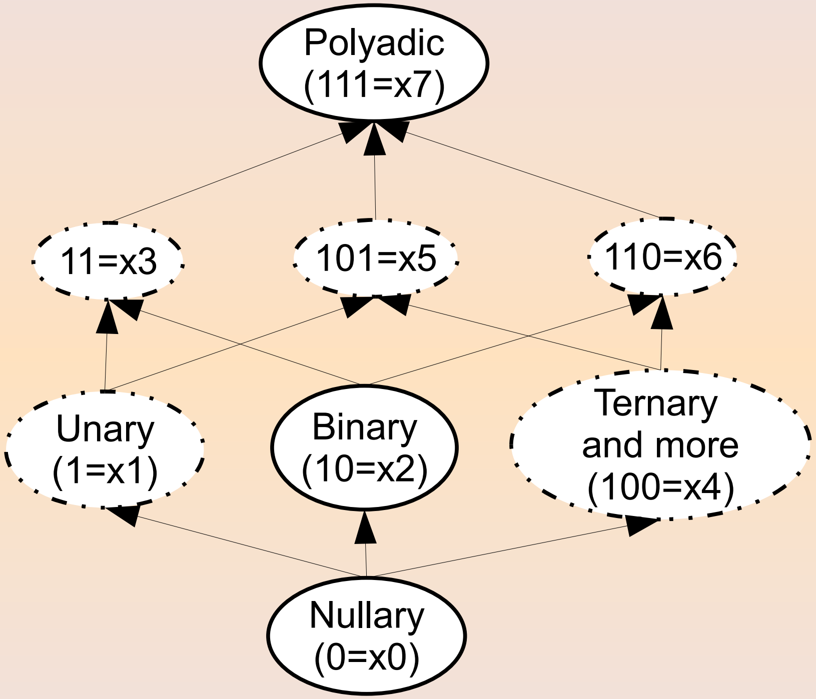


Modularization by Mix-in:
Expressive Power (backbone)


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

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**



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?)
}
```

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
```

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

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 {...}

...
```

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)

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

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**

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)