The MYNG 1.01 Suite for Deliberation RuleML 1.01: Taming the Language Lattice

Tara Athan[1], Harold Boley[2]

[1] Athan Services (athant.com), West Lafayette, Indiana, USA
taraathan AT gmail.com
[2] Faculty of Computer Science, University of New Brunswick, Fredericton, Canada
Fredericton, NB, Canada
harold.boley AT unb.ca

Abstract. This article describes the development of MYNG to Version 1.01 in order to integrate the new Deliberation RuleML Version 1.01 Relax NG schema modules – and the RuleML sublanguages they define – into the RuleML language lattice, as well as to improve the functionality of the MYNG GUI and REST interface. MYNG support is provided for including the new modules of Deliberation RuleML 1.01 into customized schemas for sublanguages such as Datalog$^+$, Hornlog$^+$, and their many extensions. To also expose Disjunctive Datalog and extensions as RuleML sublanguages, the MYNG 1.01 GUI options are better aligned with the emergent structure of Deliberation RuleML features. Together, these modifications led to a vastly increased number of RuleML sublanguages in the lattice. To assist in 'taming' this growth, we introduce the anchor lattice as an abstraction mechanism: a sublattice of the RuleML language lattice containing the most significant Deliberation RuleML sublanguages. A MYNG algorithm and interface to discover anchors are offered. For each anchor, the highly modular Relax NG schema has been automatically converted into a monolithic XSD schema, maximizing compatibility with XML tools.

1 Introduction

RuleML[1] is a knowledge representation language designed for the interchange of the major kinds of Web rules in an XML format that is uniform across various rule logics and platforms. It has broad coverage and is defined as an extensible family of sublanguages, whose modular system of XML-serialization schemas permits rule interchange with high precision. RuleML 1.0 encompasses both Deliberation RuleML 1.0[2] and Reaction RuleML 1.0[3].

[1] http://ruleml.org
[2] http://ruleml.org/1.0
[3] http://reaction.ruleml.org/1.0

Deliberation RuleML 1.0 introduced a modularization approach based on a restriction of Relax NG[AB11]. The restricted Relax NG is **monotonic**, meaning when two modules are combined, e.g. by including them both into a larger schema, the language defined by the larger schema contains both of the languages defined by the modules. Because of this monotonicity property, the more than fifty Deliberation RuleML 1.0 schema modules may be freely combined to define a fine-grained poset lattice of RuleML sublanguages, with a partial order based on syntactic language containment. In order to manage the large number of RuleML sublanguages generated by this approach, the Modular sYNtax confiGurator (MYNG) application[Ath11] was developed to provide a unified parameterized schema accessible either directly, using a REST interface, or through a GUI that exposes the REST interface.

MYNG may be used to configure a RuleML sublanguage with a set of desired features. The output of the MYNG 1.0 GUI includes:

- the MYNG REST interface URL for a Relax NG schema of the selected language, and
- a display of the configured Relax NG schema driver.

Relax NG schemas configured using MYNG 1.0 (online through the MYNG REST interface URL or as local copy after download) may be used outside of MYNG for schema-aware authoring, instance validation, or parser generation through XML tools such as oXygen XML and JAXB.

The immediate Deliberation RuleML 1.0 successor Deliberation RuleML 1.01[4] introduces several new options for obtaining an even more fine-grained customization of sublanguages, increasing the number of definable RuleML sublanguages to over 6 billion (not including serialization options). In addition to providing an interface to the larger language lattice, MYNG 1.01 introduces new functionality to assist in 'taming' its growth, including the display of more information about the Relax NG schemas and access to related XSD schemas. This paper describes the development of MYNG to Version 1.01 in order to provide this new functionality, as well as to integrate the new Relax NG schema modules, and the Deliberation RuleML Version 1.01 sublanguages they define, into the Deliberation RuleML language lattice. MYNG 1.01 has been designed and implemented to meet the following goals:

- Provide MYNG support for the new modules – and the sublanguages they define – that are introduced in Deliberation RuleML 1.01.
- Better align MYNG GUI options with the emergent structure of Deliberation RuleML features.
- Offer a MYNG algorithm and interface to discover sublanguages from the anchor sublattice, which is composed of the most significant Deliberation RuleML sublanguages.
- Improve the usability of MYNG overall.

[4] `http://deliberation.ruleml.org/1.01`

- Automate the (offline) conversion of a highly modular Relax NG schema into a monolithic XSD schema compatible with the majority of XML tools including JAXB.
- Increase the automation of the RuleML release procedure as needed for MYNG.

The rest of this article is structured as follows. Section 2 discusses the new modules of Deliberation RuleML 1.01. Section 3 expands the above goals into a design. Section 4 describes the released implementation of the design. Section 5 gives a technology roadmap. Section 6 concludes the article.

2 Overview of New Schema Modules in Deliberation 1.01

A small set of extensions of Datalog yields a major payoff: a standard XML serialization of Datalog$^+$[GOPŠ12][5], a superlanguage of the decidable Datalog$^\pm$[6]. The highlight of Deliberation RuleML 1.01 is the ability to combine one or more of the following Datalog extensions which together define Datalog$^+$:

- **Existential Rules**, where the *then* part of a rule has existentially quantified variables, as required for description logics[7], F-logic[8] and PSOA RuleML[9], Rule-Based Data Access[10], etc.
- **Equality Rules**, where the *then* part of a rule is the `Equal` predicate, as needed for user-defined/'semantic' equality in logics with equality[11] and functional logic programming[12] (this was already allowed in RuleML 1.0)
- **Integrity Rules**, where the *then* part of a rule is falsity, as a convenient way to express negative integrity constraints[13].

Note that while Datalog querying is decidable, the Datalog$^+$ extension is undecidable in its full generality. A decidable Datalog$^+$ sublanguage, such as Datalog$^\pm$, may be obtained by imposing restrictions, e.g. using Schematron (see Section 5 for a discussion of future plans for the MYNG framework.)

In Deliberation RuleML 1.01, each of these Datalog$^+$ features can now also be combined with a conjunction (e.g., an `And` of `Atom`s, rather than just one `Atom`) in the *then* part, as used, e.g., in SWRL[14]. Moreover, Deliberation RuleML 1.01

[5] `http://www.slideshare.net/polibear/datalog-and-its-extensions-for-semantic-web-databases`
[6] `http://ontolog.cim3.net/file/work/RulesReasoningLP/2013-10-31_Concepts-Foundations-I/Datalog-plus-minus_GeorgGottlob-AndreasPieris_20131031.pdf`
[7] `http://dl.kr.org/`
[8] `http://flora.sourceforge.net/`
[9] `http://wiki.ruleml.org/index.php/PSOA_RuleML`
[10] `http://wiki.ruleml.org/index.php/Rule-Based_Data_Access`
[11] `https://www.mpi-sb.mpg.de/~uwe/lehre/autreas/v13.pdf`
[12] `https://www.informatik.uni-kiel.de/~mh/FLP/`
[13] `http://www.doc.ic.ac.uk/~rak/papers/newbook.pdf`
[14] `http://www.w3.org/Submission/SWRL/`

permits mixing-in the characteristic Disjunctive Datalog feature of disjunction (e.g., an `Or` of `Atoms`) in the *then* part, as used for implementing description logics[15].

The portion of the RuleML language lattice corresponding to Datalog$^+$ features is shown in Fig. 1.

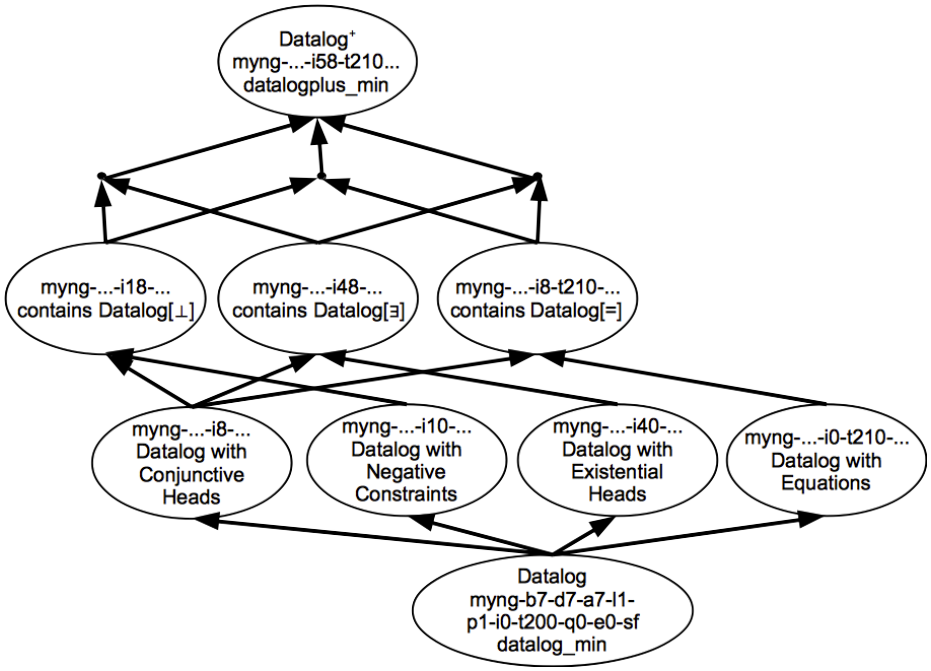

Fig. 1. Hasse diagram for a sublattice of the RuleML language lattice with infimum corresponding to the Datalog language (anchor `datalog_min`) and supremum corresponding to the Datalog$^+$ language (anchor `datalogplus_min`). All of the vertices of the sublattice are identified by their myng-codes. The myng-code for Datalog is shown in its entirety, while for other vertices only the components that differ from Datalog are shown. The first row of vertices above the infimum corresponds to individual MYNG GUI checkbox options: from left to right, *Conjunctive Heads*, *Negative Constraints*, *Existential Heads*, and *Equations*. The vertices in the second row above the infimum are labeled according to the decidable Datalog extensions they contain. The third row of vertices, unlabeled, corresponds to combinations of conjunctive heads and any two of the other three options, while the supremum, Datalog$^+$, includes all four options.

[15] `http://www.cs.ox.ac.uk/boris.motik/pubs/hms07reasoning.pdf`

An instructive suite of examples[16] demonstrates the syntax and semantics of Datalog$^+$ with annotated rules about a business scenario. We include several excerpts here.

Example 1. Negative constraints to express disjoint classes: "Nothing is both an employee and a department." (Negative-constraint rules employ `Or()` conclusions to represent falsity – "$\bot$" of Datalog[$\bot$] – in a queryable manner.)

```
<Forall><Var>x</Var>
  <Implies>
    <if>
      <And>
        <Atom>
          <Rel>employee</Rel>
          <Var>x</Var>
        </Atom>
        <Atom>
          <Rel>department</Rel>
          <Var>x</Var>
        </Atom>
      </And>
    </if>
    <then>
      <Or/>
    </then>
  </Implies>
</Forall>
```

Example 2. *Equality rules* for expressing functionality: "Everything has at most one supervisor. "

```
<Forall><Var>x</Var><Var>y</Var><Var>z</Var>
  <Implies>
    <if>
      <And>
        <Atom>
          <Rel>supervises</Rel>
          <Var>x</Var>
          <Var>z</Var>
        </Atom>
        <Atom>
          <Rel>supervises</Rel>
          <Var>y</Var>
          <Var>z</Var>
        </Atom>
      </And>
    </if>
    <then>
```

[16] `http://deliberation.ruleml.org/1.01/exa/DatalogPlus/datalogplus_min.ruleml`

```
        <Equal>
          <Var>x</Var>
          <Var>y</Var>
        </Equal>
      </then>
    </Implies>
  </Forall>
```

Example 3. *Existential rules*: "Every employee who directs a department is a manager, and supervises at least another employee who works in the same department."

```
<Forall><Var>E</Var><Var>P</Var>
  <Implies>
    <if>
      <And>
        <Atom>
          <Rel>employee</Rel>
          <Var>E</Var>
        </Atom>
        <Atom>
          <Rel>directs</Rel>
          <Var>E</Var>
          <Var>P</Var>
        </Atom>
      </And>
    </if>
    <then>
      <And>
        <Atom>
          <Rel>manager</Rel>
          <Var>E</Var>
        </Atom>
        <Exists><Var>E'</Var>
          <And>
            <Atom>
              <Rel>supervises</Rel>
              <Var>E</Var>
              <Var>E'</Var>
            </Atom>
            <Atom>
              <Rel>works_in</Rel>
              <Var>E'</Var>
              <Var>P</Var>
            </Atom>
          </And>
        </Exists>
      </And>
    </then>
  </Implies>
</Forall>
```

In addition to the new features for Datalog$^+$, described above, Deliberation RuleML 1.01 also has an expanded range of features regarding the length of term sequences. These **arity** features allow closer approximations to important Semantic Web languages: Unary (length one) term sequences in relations and functions, freely combinable with Binary (length two) term sequences, useful for graph logics (RDF[17]) and description logics (OWL[18]).

Because of the modular schema design, all of the new features of Deliberation RuleML 1.01 are freely combinable, through module inclusion, with each other and with the existing RuleML sublanguages. In particular, the new features are available for other logics in Deliberation RuleML, including Horn logic (Hornlog RuleML 1.01), e.g. allowing *Hornlog existential rules*, *Hornlog equality rules*, and *Hornlog integrity rules*, together constituting what may be called **Hornlog**$^+$.

3 Design of the MYNG 1.01 Release

The goals presented in Section 1 are revisited in the current section under the design perspective.

3.1 MYNG Support for New Deliberation RuleML 1.01 Modules

The Deliberation RuleML modular schema design was created to allow additions to the language lattice while maintaining backward compatibility. Sublanguages in the Deliberation RuleML language lattice are defined by a set of schema modules, and are identified by a myng-coded name. This unique name is a compact representation of the REST query that may be used to obtain the driver schema that combines the defining set of schema modules into one schema. For example, the lattice supremum language, containing all other Deliberation RuleML sublanguages, for Version 1.0 is called

```
myng-b3f-d7-a7-l1-p3ff-i7-tf3f-a7-ef-sf
```

and is available from the REST call (a URI with a query)
`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`[19]

The myng-code is composed of ten components – collections of modules that have some common characteristics. Each myng-code component corresponds to a GUI facet, and a REST query parameter.

[17] `http://www.w3.org/RDF/`
[18] `http://www.w3.org/OWL/`
[19] The PHP script at `http://ruleml.org/1.0/relaxng/schema_rnc.php` implements the MYNG REST interface to the parameterized schema for Deliberation RuleML 1.0.

Each GUI facet has a number of options – checkboxes or radio buttons – that, for the most part[20], correspond to single schema modules, and to single bits in the hexadecimal values in the myng-code components and REST query parameter values.

For example, in MYNG 1.0 the GUI facet labeled **Implication Options**, corresponding to the REST query parameter `implies`, and the myng-code component `i`, contains three checkboxes[21]. Two checkboxes correspond to the modules defining the values of attributes `@material`, `@direction` (and their `map` counterparts) for semantic and pragmatic variants of the RuleML implication element `<Implies>`. A third checkbox – corresponding to the module defining *if and only if* logical connective `<Equivalence>`, which is syntactic sugar for a pair of implications – also belongs to this GUI facet. The hexadecimal value of the `i` component of the myng-code `myng-b3f-d7-a7-l1-p3ff-i7-tf3f-a7-ef-sf` is 7, or 111 in binary, indicating all options have been checked.

In the MYNG 1.01 GUI[22], we extend the *Implication Options* GUI facet to include the following new options, each corresponding to a schema module that is new in Deliberation RuleML 1.01:

- **Conjunctive Heads**: Not included in Version 1.0 because conjunction in the *then* part of a Datalog rule is syntactic sugar for a pair of rules, but significant for Version 1.01 because existentially-quantified conjunctions in the *then* part of a Datalog$^+$ rule cannot always be decomposed.
- **Negative Constraints**: The empty `Or`, interpreted as constantly false, playing the role of *falsity* in RuleML, is allowed in the *then* part of an implication.
- **Existential Heads**: The existential quantifier may be arbitrarily nested with the logical connectives allowed in the *then* part of implications.

Three additional bits are prepended to the `implies` REST query parameter, and `i` myng-code component, to accommodate these three new modules.

The schema module for unary term sequences was anticipated in the Version 1.0 myng-code, so we may add options to the *Term Sequences* GUI facet without extending the range of values for the REST query parameter `termseq` (myng-code component `a` – for arity). In this case, we simply activate a bit that was present, but not used, in the Version 1.0 myng-code and REST interface.

3.2 Alignment of MYNG GUI Options with Deliberation RuleML Features

The considerations of Datalog extensions to realize the Datalog$^+$ sublanguage of RuleML led to an additional insight regarding **Disjunctive Rules**, where a non-empty `Or` may appear in the *then* part of an implication. The module

[20] A few GUI options correspond to more than one module for the sake of a more consistent design, and a few modules correspond to no GUI option because they are required in all non-vacuous sublanguages.

[21] `http://ruleml.org/1.0/myng`

[22] `http://deliberation.ruleml.org/1.01/myng`

implementing this feature is present in Version 1.0, but is exposed in MYNG 1.0 as part of the **Expressivity "Backbone"** GUI facet, a group of radio buttons where a core expressivity level is selected[23]. The Version 1.0 design parallels the hierarchical modularization of the precursor XSD schemas in this regard.

In order to implement Disjunctive Datalog and related Datalog extensions, it was necessary to detach the option for Disjunctive Heads from the *Expressivity "Backbone"* GUI facet and attach it to the *Implication Options* GUI facet, with a corresponding increase by one bit to the range of the `implies` REST query parameter and the `i` myng-code component. However, for backward compatibility with Version 1.0, a bit representing this option is also retained in Version 1.01 in the `backbone` REST query parameter and the `b` myng-code component.

3.3 Anchor Discovery

A few Deliberation RuleML sublanguages of special significance have been designated as **anchor** sublanguages (or simply anchors); XSD schemas (automatically generated in advance by offline scripts) are made available for these anchor sublanguages. The supremum language is an anchor, as are the original fifteen Deliberation RuleML languages of Version 1.0, as well as the greatest Deliberation RuleML sublanguage for each level of the *Expressivity "Backbone"*.[24] For any given Deliberation RuleML sublanguage $\mathcal{L}$, there is at least one minimal anchor sublanguage $\mathcal{L}_a$ containing it. The XSD schema for $\mathcal{L}_a$ is called an anchor XSD schema for $\mathcal{L}$.[25]

The design of the language lattice allows implementation of a simple search algorithm for an anchor schema of some Deliberation RuleML sublanguage $\mathcal{L}$ – the myng-code of $\mathcal{L}$ is compared to the myng-codes of the anchors, in ascending lexicographic order, until an upper bound is found. As long as the membership of the anchor lattice is kept to a reasonable size, this is not an expensive computation as it is linear in the size of the anchor lattice, and each comparison is linear in the size of the myng-code, roughly the number of schema modules.[26]

The Deliberation RuleML anchor lattice will remain small because the generation of the XSD schema from the modular Relax NG must be repeated for each sublanguage, due to the inherent incompatibility of the fine-grained Relax NG modularization approach with the XSD schema language. A positive outcome of this incompatibility is **schema monolithification**; the generated XSDs

[23] The available expressivity levels in MYNG 1.01 are *Atomic Formulas*, *Ground Fact*, *Ground Logic*, *Datalog*, *Horn Logic* and *Full First-Order Logic*, while MYNG 1.0 also contains *Disjunctive Logic*.

[24] The intersection of all anchors is itself an anchor, so that the lattice property is attained also for the partially ordered subset of anchors. This infimum anchor is not directly available through the MYNG GUI.

[25] The anchor(s) for any particular sublanguage is dependent on the membership of the anchor lattice, and may not be unique.

[26] In the case of a large anchor lattice, the complexity of the search could be reduced with standard techniques.

are made monolithic, hence compatible with tools that do not allow recursive redefinition, such as JAXB[27].

3.4 MYNG GUI Usability

To improve usability, the layout, styling and controls of the MYNG GUI have been modified. The ten GUI facets have been reordered so that options with the greatest semantic significance are encountered first on left-to-right, top-to-bottom traversal. Text fields have been added to display the myng-code and anchor of the form selection – this information is updated with every selection. Download buttons for the RNC and XSD anchor schemas have been added, and download is immediately available with every selection, independent of the existing *Generate Schema* functionality. The schema URLs, necessary for use of the online versions of the schemas, are now displayed below the form, including the URL for the anchor XSD schema. Like the text fields, these URLs are updated with every selection. Finally, color and font variation have been added to the GUI for an improved visual design.

3.5 RuleML Release Automation as Needed for MYNG

As in other systems, as the complexity of the RuleML language family increases, the difficulty of preparing a new release tends to also increase. However, to counteract this trend, in Version 1.01 we have made some strides towards automating the release process, including the following new or improved procedures:

- Version control of schemas, MYNG code, and associated artifacts.
- Automatic publishing to the RuleML server from the version control system.
- Automation of the conversion from modular Relax NG to monolithic XSD.
- Batch generation of anchor XSD schemas.
- An improved Normalizer.

4 Implementation of the MYNG 1.01 Release

The design presented in Section 3 is taken up in the current section for describing the implementation. A snapshot of the MYNG GUI is shown in Fig. 2. Tutorials for using the MYNG GUI to create a custom Deliberation RuleML 1.01 sublanguage, usage of the MYNG REST interface, and documentation for contributors to the development of RuleML are available on the RuleML Wiki.[28]

The following subsections are organized according to the implemented artifacts, some spanning multiple design aspects.

[27] The manually-written modular XSDs of Deliberation RuleML 1.0 rely on recursive redefinition for their hierarchical modularization approach, hence are not JAXB-compatible.

[28] `http://wiki.ruleml.org/index.php/MYNG_of_Deliberation_RuleML_1.01_Demo`

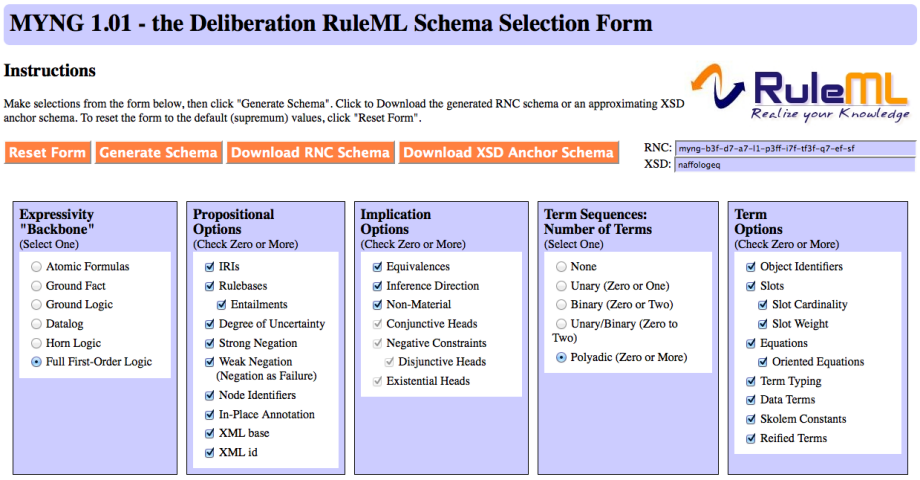

Fig. 2. The top portion of the MYNG GUI includes brief instructions, a set of buttons for managing the form and downloading schemas, and text fields displaying the myng-code and anchor. The figure shows the first row of five GUI facets – the second row is not shown for space considerations. The Relax NG schema URL and the driver schema are displayed below the second row of GUI facets, in addition to some more detailed instructions on usage of the schema.

4.1 MYNG 1.01 GUI

The MYNG GUI has been implemented as an XHTML form with embedded JavaScript since Version 1.0. AJAX is used to call the MYNG REST interface, with query parameters based on the user's selections in the form. To accommodate the new Version 1.01 schema options and controls, the JavaScript has been extensively refactored, producing a cleaner code base and allowing future extensions to be made more easily.

4.2 Parameterized Relax NG Schema

The Deliberation RuleML parameterized Relax NG schema has been implemented as the MYNG REST interface calling a PHP script since Version 1.0. In MYNG 1.01, a minor extension has been implemented, allowing an increased range for the `implies` REST query parameter, for the new Datalog$^+$ modules, and activating intermediate values of the `termseq` REST query parameters, for the new unary term sequences module.

4.3 Release Automation Scripts

A set of open-source Bash scripts[29] has been developed for Deliberation RuleML 1.01 to carry out the first stage of the batch generation of anchor schemas,

[29] `https://github.com/RuleML/deliberation-ruleml/tree/1.01-dev/bash`

utilizing Jing-Trang[30]. These scripts download the appropriate driver Relax NG schema for each anchor from the MYNG REST interface running on the RuleML server, simplify the schema using Jing, and then convert it to XSD using Trang.

The batch script (`batch_web2xsd.sh`) calls an auxiliary script (`aux_web2xsd.sh`) that executes the conversion of a single modular Relax NG to monolithic XSD. The auxiliary script may be used directly to obtain XSD schemas for RuleML sublanguages other than the anchors.

4.4 XSLT Transformation for Post-Conversion XSD Processing

An XSLT transformation has been developed for Deliberation RuleML 1.01 to implement the second (and final) stage of the generation of monolithic XSD schemas, including the anchor XSD schemas. This transformation handles some bugs in the Trang-based conversion process, as well as dealing with some inherent incompatibilities between Relax NG and XSD.

The most significant incompatibility is in regard to the `@xsi:type` attribute. XSD schemas implicitly allow this attribute to be used on any element, permitting users to modify the content model. Relax NG has no such mechanism – to use the `@xsi:type` attribute, its behavior must be defined in the grammar. XSD requires the value of `@xsi:type` to be a validly-derived type of the element type declared in the schema, inhibiting the mixing of attributes on the `<Data>` element with explicit datatype declaration through `@xsi:type`. To work around this constraint, we define a set of types[31] in the RuleML namespace to reflect the XML Schema Datatypes. For example, the following fragment is valid against both Relax NG and XSD schemas:

```
<ruleml:Data node="#d1" xsi:type="ruleml:integer">123</ruleml:Data>
```

while the following fragment is not:

```
<ruleml:Data node="#d1" xsi:type="ruleml:integer">abc</ruleml:Data>
```

After the transformation is applied to all anchor XSD schemas, the schemas must be committed to the Github repository for RuleML[32], which automatically publishes onto the RuleML server through a cron job.

4.5 XSLT Transformations for Normalization

The Deliberation RuleML Normalizer has been implemented since Version 1.0 as an XSLT transformation that converts a Deliberation RuleML instance file in the relaxed-form serialization[33] into the normalized serialization. In Version 1.01, the

[30] `https://code.google.com/p/jing-trang/`

[31] `http://deliberation.ruleml.org/1.01/datatypes/SimpleWithAttributes.xsd`

[32] `https://github.com/RuleML/deliberation-ruleml`

[33] RuleML has long allowed an abbreviated serialization (with optionally skipped edge tags, also called **stripes**, and default attribute values) and some freedom in the ordering of elements. From Version 1.0 and up, Relax NG schemas for the **relaxed serialization** have extended the abbreviated serialization to allow even more freedom in element ordering than the XSD schemas, with all permutations possible as long as any missing edge tags or attributes can be unambiguously reconstructed.

Normalizer has been fixed to provide the correct attributes with default values for all RuleML sublanguages.[34] It is expected that default values of attributes will no longer be used in Deliberation RuleML 1.02, which, among other postive benefits, would allow a single Normalizer XSLT to be used for all sublanguages.

5 Roadmap of MYNG 1.02 and Beyond

The RuleML Wiki hosts the RuleML issue tracking system[35], which is used to manage the development of MYNG, as well as the RuleML language itself. Users are encouraged to post bug reports and enhancement requests. In MYNG Version 1.02 and beyond, we expect to see the following improvements.

- New syntax from Deliberation RuleML 1.02 will be integrated into MYNG.
- New anchor languages will be added, based on user suggestions.
- MYNG functionality will be improved, including:
  - Add a *Clear* button[36] that unchecks all checkboxes, to facilitate configuration of minimal RuleML sublanguages, as a complement to the current *Reset* button that favors maximal sublanguages.
  - Provide a visual indicator of the (lattice) distance from the anchor to the RuleML sublanguage selected by GUI options, especially the zero-distance case of exact match between the anchor and the selected options.
  - Implement the inverse functionality of determining the options corresponding to a given anchor or named language selected by the user.
- Progress will continue towards making MYNG a *language-lattice development environment*:
  - Add validation of an instance against the configured Relax NG schema, using, e.g., Validator.nu.[37]
  - Implement the determination of the lub schema for a valid instance of the supremum language, using validation against *penultimate* Relax NG schemas containing all modules but one, and properties of the language lattice.[38]
  - Add Schematron validation, e.g. using Validator.nu, to impose additional language restrictions, such as the Datalog$^{\pm}$ restrictions that restore decidability
  - Enable on-the-fly conversion of Relax NG schemas configured using MYNG into monolothic XSD schemas

[34] `http://deliberation.ruleml.org/1.01/xslt/normalizer/`
[35] `http://wiki.ruleml.org/index.php/Category:Issues`
[36] `http://wiki.ruleml.org/index.php/MYNG_GUI_Clear_Button`
[37] `http://validator.nu`
[38] `http://wiki.ruleml.org/index.php/MYNG_Checker`

6 Conclusions

MYNG has been envisioned as a framework for the design and use of highly-configurable syntaxes from a language lattice, specified with modular schemas expressed in a monotonic restriction of Relax NG. Following up on the prototype MYNG 1.0, here we have described the goals, design and implementation of MYNG 1.01, and provided a roadmap to MYNG 1.02 and beyond. The MYNG schema design pattern was initially applied to Deliberation RuleML (1.0, and then 1.01) and recently transferred to Reaction RuleML (1.0). We look forward to integrating MYNG with other XML applications.

References

[AB11] Tara Athan and Harold Boley. Design and implementation of highly modular schemas for xml: Customization of ruleml in relax ng. In Frank Olken, Monica Palmirani, and Davide Sottara, editors, *Rule - Based Modeling and Computing on the Semantic Web*, volume 7018 of *Lecture Notes in Computer Science*, pages 17–32. Springer Berlin Heidelberg, 2011.

[Ath11] Tara Athan. Myng: Validation with ruleml 1.0 parameterized relax ng schemas. In *Proceedings of the 5th International RuleML2011@BRF Challenge, co-located with the 5th International Rule Symposium*. CEUR-WS.org, 2011.

[GOPŠ12] Georg Gottlob, Giorgio Orsi, Andreas Pieris, and Mantas Šimkus. Datalog and its extensions for semantic web databases. In Thomas Eiter and Thomas Krennwallner, editors, *Reasoning Web. Semantic Technologies for Advanced Query Answering*, volume 7487 of *Lecture Notes in Computer Science*, pages 54–77. Springer Berlin Heidelberg, 2012.