MYNG: Validation with RuleML 1.0 Parameterized Relax NG Schemas

Tara Athan

Athan Services, Ukiah, CA, USA
`taraathan AT gmail.com`

Abstract. The knowledge representation language RuleML Version 1.0 has recently been re-engineered using the Relax NG schema language, introducing several new features, including on-the-fly schemas with fine-grained, freely-combinable modules. The web application Modular sYNtax confiGurator (MYNG) provides GUI access to a PHP-based parameterized schema. To ensure monotonicity when combined, the modules follow a schema design pattern that is enforced by a meta-schema. The schema design pattern also facilitates user-extension of the language. The usage of these new features of RuleML are demonstrated at the website `http://wiki.ruleml.org/index.php/MYNG#Demo` using H. Sivonen's online, open source validator, `http://Validator.nu`.

1 Introduction

The knowledge representation language RuleML Version 1.0 has recently been re-engineered using the Relax NG schema language [ABss]. The Relax NG schema language is known to be more expressive than XSD or DTD [Gen07]. It has also been shown to be more efficient for XML validation [SK07], where benchmark studies comparing the performance of the Jing validator (against Relax NG schemas) to the Xerces and MSV validators (against XSD schemas) showed reductions by 25% to 50% in the time required for validation, (scaling linearly with file length).

Although validating against Relax NG schemas has advantages over the XML Schema Definition Language (XSD) schemas in many respects, Relax NG is not as widely used as XSD. While some commercial XML editors, notably oXygen[1] support Relax NG schemas, some other popular editors do not. Therefore, a large segment of the population of current and prospective RuleML users is, in all likelihood, unfamiliar with validation using Relax NG.

Further, the re-engineered RuleML Relax NG schemas introduce several new features, including

– a PHP-driven parameterized schema driver that delivers a customized schema on-the-fly based on query string parameters in the schema URL;
– a schema design pattern that ensures monotonicity when schema modules are freely combined.

[1] oXygen: `http://oxygenxml.com`

Advanced users who wish to fully exploit the customizability and extensibility of the RuleML Relax NG schemas require a thorough understanding of these features.

Therefore we present a demonstration of validation using RuleML 1.0 schemas, beginning with the simplest case based on validation of a pure RuleML instance using a redirected link, progressing to validation of mixed-namespace instances using customized and user-extended schemas, and finishing with validation of Relax NG schemas against a meta-schema defining a schema design pattern.

2 Overview of Validator.nu

For this demo, we will make use of Henri Sivonen's Validator.nu[2], a free validation webservice [Siv07] that can validate an XML instance against schemas, including Relax NG schemas and Namespace-Based Validation Dispatching Language[3]. The validation engine used by Validator.nu is Jing[4], an opensource application with command-line interface developed by James Clark, one of the authors of Relax NG [ISO08] itself.

Validator.nu has a simple user interface, allowing the user to specify a single instance document and zero to many schemas. Options include namespace-based filtering, allowing a particular namespace to be ignored by the validator.

3 Examples

The following Validator.nu cases may be accessed via links from the RuleML wiki[5].

3.1 Redirected Links

Example 1 demonstrates an attempt to validate a test RuleML instance in the `bindatagroundlog` sublanguage with the `bindatagroundfact` relaxed schema[6], accessed via a redirected link to the parameterized schema. As expected, the validator finds errors.

The actual URL that is accessed in this example is `http://www.ruleml.org/1.0/relaxng/schema_rnc.php?backbone=x1&default=x7&termseq=x2&lng=x1&propo=xf&implies=x6&terms=xf0f&quant=x1&expr=x0&serial=xf`. Redirections to the parameterized schema have been implemented for the original fifteen named

[2] Validator.nu: `http://validator.nu`

[3] NVDL: `http://nvdl.org`

[4] Jing: `http://code.google.com/p/jing-trang/`

[5] Validator.nu Links:`http://wiki.ruleml.org/index.php/MYNG#Validator.nu_Examples`

[6] See `http://ruleml.org/1.0/relaxng/bindatagroundfact_relaxed.rnc` for the most inclusive schema.

RuleML sublanguages in the Deliberation family (except for the SWSL languages), from `bindatagroundfact` to `naffologeq`, in both serializations (normal and relaxed form). A complete listing of these redirects is available at the website `http://ruleml.org/1.0/relaxng/`.

3.2 Direct Links

In example 2, we use a direct link to the PHP-driven parameterized schema for validating a RuleML instance with a foreign namespace element. Any elements or attributes whose names belong to the foreign namespace are ignored by the validator. This mode of validation accepts RuleML that is emebedded in other documents.

3.3 NVDL

Example 3 shows the validation of a RuleML instance against an NVDL script that refers to the parameterized schema and also allows arbitrary elements from foreign namespaces. The NVDL script is:

```
<?xml version="1.0" encoding="UTF-8"?>
<rules xmlns="http://purl.oclc.org/dsdl/nvdl/ns/structure/1.0"
  xmlns:a="http://relaxng.org/ns/compatibility/annotations/1.0">
  <namespace ns="http://www.ruleml.org/0.91/xsd">
    <validate
      schema="http://ruleml.org/0.91/relaxng/schema_rnc.php?backbone=x3f&default=x7&
termseq=x7&lng=x1&propo=x3f&implies=x7&terms=xf3f&quant=x7&expr=xf&serial=xf"
      schemaType="application/relax-ng-compact-syntax"/>
  </namespace>
  <anyNamespace>
    <allow/>
  </anyNamespace>
</rules>
```

3.4 Static Schema

There are several reasons why a user may want to validate against a static copy of the RuleML schema, including performance and offline operation. Example 4 demonstrates validation of a RuleML instance against a static copy of the default output of the parameterized schema. The schema may be obtained by two methods:

– use the MYNG GUI[7] to display a direct link to the PHP script, click on the link, and save the output to a file;
– scrape the schema driver displayed on the RuleML MYNG GUI;
– download one of the zip archives, built on-demand for either normal and relaxed form, (see `http://ruleml.org/1.0/relaxng/` and extract the schema driver file for any of the named sublanguages;

In all cases, the directory containing the schema driver must also contain the module directory, which is included in both of the zip archives.

[7] MYNG GUI: `http://ruleml.org/1.0/myng`

3.5 Including Extensions

Users may wish to extend the RuleML syntax, and the Relax NG modular schema was designed to make such extensions as convenient as possible. As a template for such extensions, we show in example 5 the expansion of the RuleML parameterized schema by a Boolean operator for exclusive disjunction.

Four modules were written to implement this extension:

- the definition module[8] contains the definition of new elements;
- the stripe-skipping module[9] contains the code that allows redundant edge elements to be skipped;
- the dishornlog expressivity module[10] contains the code that makes the extended schema compatible with languages having
- the folog expressivity module[11] contains the code that makes the extended schema compatible with languages having at least the expressive power of first-order logic.

Other modules might be required for some extensions, including:

- specification of different content models for normal and relaxed serializations;
- specification of different content models for attributes with default values;
- additional expressivity modules if content models change with other levels of expressivity;

The schema driver contains the following include statements:

```
namespace rulemlx = "http://www.ruleml.org/0.91/ext"
include "http://ruleml.org/0.91/relaxng/schema_rnc.php"
include "http://ruleml.org/0.91/relaxng/modules-ext/xor_expansion_module.rnc"
  inherit = rulemlx {start |= notAllowed}
include "http://ruleml.org/0.91/relaxng/modules-ext/xor_stripe_skipping_expansion_module.rnc"
  inherit = rulemlx {start |= notAllowed}
include "http://ruleml.org/0.91/relaxng/modules-ext/xor_dis_expansion_module.rnc"
  inherit = rulemlx {start |= notAllowed}
include "http://ruleml.org/0.91/relaxng/modules-ext/xor_fo_expansion_module.rnc"
  inherit = rulemlx {start |= notAllowed}
```

The first statement includes the RuleML language `naffologeq` with the relaxed serialziation. The second statement includes the required expansion module for the Xor element. The other three statements include optional expansion modules that

- allow the `formula` edge to be skipped;
- allow the Xor element to appear in the conclusions of implications;
- allow the Xor element to appear in rulebase assertions and retractions.

[8] `http://ruleml.org/1.0/relaxng/modules-ext/xor_expansion_module.rnc`

[9] `http://ruleml.org/0.91/relaxng/modules-ext/xor_stripe_skipping_expansion_module.rnc`

[10] `http://ruleml.org/1.0/relaxng/modules-ext/xor_dis_expansion_module.rnc`

[11] `http://ruleml.org/1.0/relaxng/modules-ext/xor_fo_expansion_module.rnc`

3.6 In xhtml

In keeping with the original purpose of Validator.nu, which is (X)HTML5 validation, we demonstrate the validation of RuleML that is embedded in the `header` section of an xhtml document in Example 6. NVDL is used to validate against three schemas, the xhtml Relax NG schema, the xhtml Schematron restrictions and a RuleML schema.

3.7 RNG Schema Validation

In [ABss], a schema design pattern was introduced that ensures monotonicity of the language when modules are freely mixed. Validator.nu can be used to validate a schema in the XML-based Relax NG syntax (RNG) against the meta-schema, also in the RNG syntax, that defines the schema design pattern. The meta-schema includes and redefines the standard RuleML schema[12], restricting the vocabulary of named patterns to three categories based on their suffixes:

- Choice combine elements: with suffixes

  `( choice | main | content | value | datatype | sequence | defs )`
- Interleave combine elements: with suffixes

  `( attlist | header | notallowed )`
- No combine elements: with suffix

  `( def )`

Monotonicity is achieved by restricting patterns in the interleave combine category to be optional [ABss].

3.8 Performance

The greatest part of the execution time of these examples is spent on remote access of the schemas and instance. Thus any XML validation with serious concerns about performance must utilize local copies of the validator and schemas, or use caching, a feature not available in Validator.nu.

4 Conclusion

We have demonstrated a number of examples of instance validation using the RuleML Relax NG schemas, ranging from simple cases to multi-namespace instances, and customized, user-extended schemas. We have also shown how a schema in the XML-based Relax NG syntax may be validated against a custom schema that enforces a schema design pattern.

Because of the limitations of the Validation.nu webservice, there are several uses of these schemas that we are not able to demonstrate here, including

[12] Standard Relax NG schema in RNG: `http://relaxng.org/relaxng.rng`

– validation of a schema in the compact Relax NG syntax
– conversion between schema languages (Relax NG compact, XML-based, XSD)
– conversion of a modular Relax NG schema into a simplified monolithic schema
– generation of an XML parser from a Relax NG schema

All of these tasks can be accomplished on the desktop using opensource software[13] and all but the last are available in the commercial oXygen framework. Future work includes the development of Java Webstart services that provide these additional capabilities.

References

[ABss] Tara Athan and H. Boley. Design and implementation of highly modular schemas for xml: Customization of ruleml in relax ng. In F. Olken, M. Palmirani, and D. Sottara, editors, *Rule-Based Modeling and Computing on the Semantic Web*. RuleML 2011 - America, LNCS 7018, in press.

[Gen07] Pierre Geneves. Logics for xml. `http://hal.inria.fr/docs/00/13/35/91/PDF/geneves-phd.pdf`, 2007.

[ISO08] ISO. ISO/IEC 19757-2: Document Schema Definition Language (DSDL) Part 2: Regular-grammar-based validation - RELAX NG. `http://standards.iso.org/ittf/PubliclyAvailableStandards/c052348_ISO_IEC_19757-2_2008(E).zip`, 2008.

[Siv07] Henri Sivonen. About validator.nu. `http://about.validator.nu`, 2007.

[SK07] Mikko Saesmaa and Pekka Kilpelinen. On-the-fly validation of xml markup languages using off-the-shelf tools. `http://conferences.idealliance.org/extreme/html/2007/Saesmaa01/EML2007Saesmaa01.html`, 2007.

[13] See `http://wiki.ruleml.org/index.php/MYNG#Tools`