XMLmind XML Editor - Support of RELAX NG Schemas

Hussein Shafie, XMLmind Software <xmleditor-support@xmlmind.com>

February 22, 2019

Abstract

This document describes how RELAX NG schemas are supported by XMLmind XML Editor.

Table of Contents

1. Implementation of RELAX NG in XMLmind XML Editor

The implementation of RELAX NG in XMLmind XML Editor (XXE for short) is based on Jing, an *OpenSource*, *industrial strength*, *streaming* validator written by James Clark.

The trimmed version of Jing included in XXE (`relaxng.jar`) is used

- to load and validate RELAX NG schemas associated to XML documents (XML and compact syntaxes are both supported);
- to fully validate XML documents conforming to RELAX NG schemas, each time these documents are opened and saved, and each time a full validation is explicitly requested by the user (command Tools → Check Validity).

Jing is not used to implement guided editing. That is, Jing is not used to determine the content model of the element being edited. A quick, incremental, version of the algorithm that computes the *derivative of a pattern* is used for that.

The implementation of W3C XML Schema Datatypes used in RELAX NG schemas (e.g. `xsd:int`) is the work of XMLmind. This implementation is very different from the implementation of W3C XML Schema Datatypes included in the original Jing. This implementation is shared by the version of Jing included in XXE and by our own W3C XML Schema validator.

XXE supports attribute default values as specified in RELAX NG DTD Compatibility. The compatibility of the schema with this feature is (*very strictly*) checked by XXE and not by Jing. This means that a schema found valid by Jing but improperly using this feature will be rejected by XXE.

2. Specifying which RELAX NG schema to use for validating a document

2.1. The `relaxng` configuration element

This section is just a primer. The reference documentation about this topic is really Section 23, “relaxng” in *XMLmind XML Editor - Configuration and Deployment*.

A document type declaration (`<!DOCTYPE>`) can be used to associate a DTD to a document. Attributes `xsi:schemaLocation`/`xsi:noNamespaceSchemaLocation` can be used to associate W3C XML Schemas to a document. But there is no standard way to associate a RELAX NG schema to a document. Therefore this association must be made using an external specification such as the Namespace Routing Language (NRL).

In the case of XMLmind XML Editor, this external specification is simply a configuration element called `relaxng`.

XHTML example:

```
<configuration name="XHTML Strict [RELAX NG]"
  xmlns:html="http://www.w3.org/1999/xhtml"
  xmlns="http://www.xmlmind.com/xmleditor/schema/configuration">
  <include location="xxe-config:schema/ns_xhtml.incl" />

  <detect>
    <rootElementNamespace>http://www.w3.org/1999/xhtml</rootElementNamespace>
  </detect>

  <relaxng location="xxe-config:common/rng/xhtml1/xhtml-strict.rng" />1

  <preserveSpace elements="html:pre html:style html:script" />2

  <css name="XHTML" location="xhtml_rng.css" />
  <template name="Page" location="page.html" />
</configuration>
```

1 The `relaxng` configuration element specifies the location of the RELAX NG schema (XML syntax or compact syntax) to which conforms the document being opened.

2 Unlike the DTD, `xhtml-strict.rng` does not specify a `preserve` default value for attribute `xml:space` of elements such as `pre`. Therefore, the `preserveSpace` configuration element must be used to specify whitespace-preserving elements. More information in Section 22, “preserveSpace” in *XMLmind XML Editor - Configuration and Deployment*.

2.2. The `<?xml-model>` processing instruction

The `<?xml-model>` processing instruction allows to associate schema documents written in any schema definition language with a given XML document. As such, it may be used to associate a RELAX NG schema, written using either the XML or the compact syntax, with a document. Example (excerpts from name.xml):

```
<?xml version="1.0"?>
<?xml-model href="name.rnc" type="application/relax-ng-compact-syntax"?>
<names>
  <name><fullName>John Smith</fullName></name>
  ...
```

3. XMLmind XML Editor-friendly content models

Validating a document against a RELAX NG schema is similar to matching some text against a regular expression. If the document ``matches'' the schema, the document is valid, and this, no matter which sub-expressions were used during the match.

Example: string "`b`" matches regular expression "`(a?,b)|(b,c?)`" and we don't care if it matches sub-expression "`(a?,b)`" or sub-expression "`(b,c?)`". The situation is exactly the same with RELAX NG schemas, simply replace the characters and the character classes used in a regular expression by RELAX NG patterns.

The job of a RELAX NG schema is a validate a document as a whole, and that's it. For XXE, the problem to solve is different. One of the main jobs of XXE is to guide the user when she/he edits an XML document. That is, one of the main jobs of XXE is to identify the content model of the element which is being edited, in order to suggest the right attributes and the right child elements for it.

To do that, XXE needs to know precisely which ``sub-expressions were used during the match''. Unfortunately, sometimes, this is impossible to do.

All examples used in this section are found in *XXE_install_dir*/doc/rngsupport/samples/. Note that they are all valid schemas and valid documents.

Example 1. Ambiguous elements

RELAX NG schema, target.rnc:

```
start = build-element

build-element = element build {
    target-element*
}
target-element = element target {
    attribute name { xsd:ID },
    element list { ref-element* }?,
    element list { action-element* }?
}
ref-element = element ref {
    attribute name { xsd:IDREF }
}
action-element = element action { text }
```

Document conforming to the above schema, target_bad.xml:

```
<build>
  <target name="all">
    <list>
    </list>
  </target>

  <target name="compile"/>
  <target name="link"/>
</build>
```

If you open `target_bad.xml` in XXE and select the `list` element, XXE randomly chooses one of the two `list` content models. This is correct because both `list` content models are fine in the case of an empty `list` element. However there is a drawback: if XXE chooses the kind of `list` which contain `action` child elements, you'll have no way to insert `ref` child elements in an empty list. *In other words, one content model hides the other one.*

Now, if you open target_good.xml in XXE, there is no problem at all:

```
<build>
  <target name="all">
    <list>
      <ref name="compile"/>
      <ref name="link"/>
    </list>
    <list>
      <action>cc -c *.c</action>
      <action>cc *.o</action>
    </list>
  </target>

  <target name="compile"/>
  <target name="link"/>
</build>
```

However, in the vast majority of realistic cases, XXE knows how to make a difference between two child elements having the same name and having different content models.

RELAX NG schema, sect.rnc:

```
start = doc-element

doc-element = element doc {
    (simple-sect|
     recursive-sect)+
```

```
}

simple-sect = element sect {
    attribute class {"simple"}, paragraph-element*
}

recursive-sect = element sect {
    attribute class {"recursive"}?, (recursive-sect|simple-sect)*
}

paragraph-element = element paragraph { text }
```

Document conforming to the above schema, sect.xml:

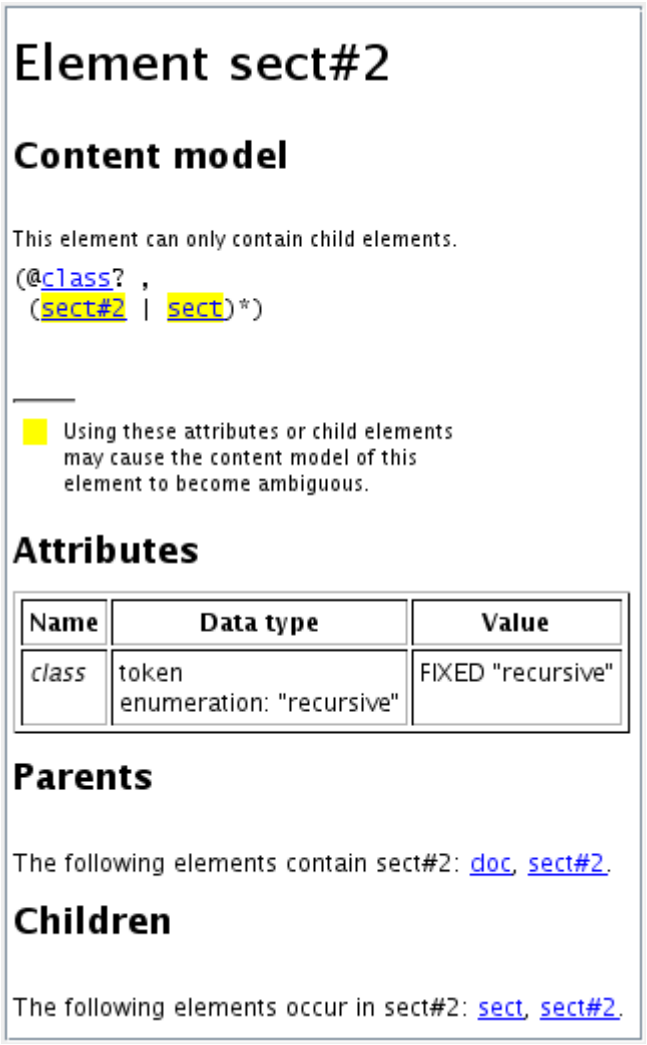
```
<doc>
  <sect>
    <sect></sect>

    <sect class="simple">
      <paragraph>Paragraph 2.</paragraph>
    </sect>
  </sect>

  <sect class="simple"></sect>
</doc>
```

In the above example, XXE has no problem at all making a difference between the empty `<sect>` element and the empty `<sect class="simple">` element. The reason is obviously because the first kind of `sect` element has a required attribute `class` with `simple` as its fixed value.

Help provided by the "Show Content Model" window

When you ask yourself “what is the content model of the (explicitly or implicitly) selected element?”, simply select menu item Help → Show Content model (keyboard shortcut: Shift+**F1**) and you'll have your answer.

Figure 1. `Sect.xml` example when first `sect` element is selected



3.1. Other content models which are not XXE-friendly

Example 2. Not specific to RELAX NG

RELAX NG schema, name.rnc:

```
start = names-element

names-element = element names {
    name-element+
}
name-element = element name {
    element fullName { text } |
    (element firstName { text } & element lastName { text })
}
```

Document conforming to the above schema, name.xml:

```
<names>
  <name><fullName>John Smith</fullName></name>

  <name><firstName>John</firstName><lastName>Smith</lastName></name>

  <name><lastName>Smith</lastName><firstName>John</firstName></name>
</names>
```

XXE allows to replace the `firstName`, `lastName` pair by a `fullName`. Simply select both child elements and use command Edit → Replace. But it is impossible to replace a `fullName` by a `firstName`, `lastName` pair.

The only way to do this is to select the `fullName` to be replaced and then, to use command Edit → Force Deletion[1]. This will force XXE to enter the so-called “lenient” editing mode. Suffice to remember that in this mode, the user is not guided. The user may add or remove any child elements she/he wants, including a `firstName`, `lastName` pair[2].

Note that the above example is not specific to RELAX NG. It is possible to model this kind of content with a DTD or a W3C XML Schema.

The example below is very similar but can only be expressed using a RELAX NG schema. This is the case, because, unlike a DTD and a W3C XML Schema, a RELAX NG schema can be used to specify the places within an element where text nodes may occur.

Example 3. Specific to RELAX NG

RELAX NG schema, name2.rnc:

```
start = names-element

names-element = element names {
    name-element+
}
name-element = element name {
    text |
    (element firstName { text } & element lastName { text })
}
```

Document conforming to the above schema, name2.xml:

```
<names>
  <name>John Smith</name>

  <name><firstName>John</firstName><lastName>Smith</lastName></name>

  <name><lastName>Smith</lastName><firstName>John</firstName></name>
</names>
```

The situation is worse with the name2.rnc example than with the name.rnc example. It is always allowed to delete a text node and this includes the text node containing "`John Smith`". That is, there is no way to force XXE to enter its “lenient” editing mode in order to be able to replace text node "`John Smith`" by a `firstName`, `lastName` pair.

In such case, using named element templates is the only way to cope with such content models. Simply specify two named element templates for element `name`, one containing a text node with a placeholder string and the other containing a `firstName`, `lastName` pair.

4. Known problems

Known problems:

- `include` and `externalRef` elements in RELAX NG schemas are XML catalog aware, but the equivalent notations in the compact syntax are not XML catalog aware.

[1] This menu item is available only after you check "Enable the 'Edit|Force Deletion' menu item" in Options → Preferences, General|Features section.
[2] The right approach here is to define two *named element templates* for element `name`, one containing a `fullName` child element and the other containing a `firstName`, `lastName` pair.